handheld competence
Jochen Hammer

palm OS CERTIFIED DEVELOPER

GPS Bluetooth Driver

Extension for Palm OS©

Disclaimer

Handheld Competence
Dipl.-Ing. Univ. Jochen Hammer

Handelstrasse 7, D-91166 Georgensgmuend
Germany

Phone: +49-9172 / 700 - 672
Fax: +49-9172 / 700 - 673
Email: info@handheld-competence.de
Web: http://www.handheld-competence.de

Version History

Version	Date	Changes
V1.0	2004-02-28	Initial release
V1.1	2004-07-09	Internal modifications to shorten the delay time in the receive buffer.

1. Abstract

The GPS Bluetooth driver for Palm OS© is an extension for the Rapid Application Tool **Mobile App Designer** from Intellisync© Inc. It allows to develop applications for Palm OS© with GPS functionality using GPS receivers with Bluetooth Wireless Technology. Without the need for establishing and maintaining the Bluetooth connection between a Bluetooth equipped Palm OS© device and a GPS Bluetooth receiver, the developer can extend existing applications or develop new applications which offer GPS navigation functionality to the user.

This driver offers all the required functionality for navigation purposes using GPS Bluetooth receivers:

- Discover Bluetooth Devices from within your application
- Connect to selected Bluetooth GPS receiver
- Check the status of the Bluetooth connection and the GPS receiver
- Receive NMEA-0183 GPS data from the GPS device
- Extract required data from the received GPS data string
- Disconnect from Bluetooth device

The driver fully supports the software 'BtToggle Pro' (www.whizoo.com). BtToggle Pro gives you complete control over your Bluetooth radio, enabling you to keep it off until you really need it, and turning it off again the moment you don't. You are also able to turn Bluetooth on and off manually with a simple press of a button.

This driver has been developed using the GPS Bluetooth receiver displayed in the picture below.

More information about Bluetooth GPS receivers can be found under:
www.socketcom.com
www.emtac.com

2. Content

3. Introduction

The GPS Buetooth driver is not using the Palm OS© virtual serial driver for a Bluetooth connection in background mode, but has be completely rewritten to use the Bluetooth library in foreground mode instead. The highly-optimized code uses less dynamic memory overall and has a small footprint. Compared to the virtual serial driver from Palmsource it is not blocking, but receiving the GPS data interrupt driven, so the User can continue using the application without any delays whilst receiving the GPS data. Any received data is parsed automatically for NMEA-0183 messages and the checksum is automatically calculated and verified once a valid GPS data string has been received. Only valid NMEA-0183 data strings are passed to the calling application.

Once a Bluetooth connection to a Bluetooth GPS receiver is established the NMEA-0183 data messages can be read from the internal receive buffer with a single function call. If the receive buffer is empty it get filled up automatically with data, which has been received via the Bluetooth interface in the meantime.

Please note, that most GPS Bluetooth receivers transmit automatically NMEA-0183 messages to the connected device. There is no need to request a GPS message, but all supported NMEA-0183 messages are send in turn by the GPS receiver. For instance the GPS receiver, which has been used during development sends the four NMEA-0183 messages: $GPRMC, $GPGGA, $GPGSV and $GPGSA in turn. Other NMEA messages, which are defined in the NMEA-0183 specification are not supported. Also note, that not each message contains all the data, which is specified in the NMEA-0183 protocol. Depending on your receiver, some data is not available. Please contact the GPS receiver manufacturer for details about the supported NMEA-0183 messages and the supported data within the message.

See the demo application for more details about implementing GPS navigation functionality in your application.

4. API Reference

The GPS Bluetooth driver supports the following 12 functions, which are described in detail below:

1. BT_Discover
2. BT_Connect
3. BT_Disconnect
4. BT_GetDeviceAddress
5. BT_SetDeviceAddress
6. BT_ReadNMEAString
7. BT_ReadNMEAData
8. BT_FlushReceiveBuffer
9. BT_ExtractParameter
10. BT_GetState
11. BT_GetDeviceName
12. BT_About

4.1 BT_Discover

Starts discovery process on Palm OS© handheld for searching Bluetooth devices in range. If Bluetooth is disabled on the Palm OS© Handheld the User is asked to switch Bluetooth on, before the discovery process continues. If the discovery process was successful and the user has selected one of the discovered Bluetooth devices in range, the address of the selected devices is stored internally. The function *BT_GetDeviceAddress* can be used to request this address. Additionally the function *BT_GetDeviceName* may be called to receive the name of the selected device as a string.

Usage:

```
result  =  BT_Discover()
```

Returns:

See result list

4.2 BT_Connect

Connect to the GPS receiver with the address, which has been received automatically during a discovery process or which has been set manually by using *BT_SetDeviceAddress*.

Usage:

```
result = BT_Connect(MessageBoxOn)
```

Returns:

See result list

Passing the boolean value '*MessageBoxOn'* results in:

False	No pop-up Messages will be created to notify the user about the connection status. (DEFAULT)
True	Pop-up Messages will be created by the driver to notify the user about the status of the connection: "GPS Bluetooth connection established" "GPS Bluetooth connection lost" "GPS Bluetooth connection dropped" These Messages boxes have to be confirmed from the user by pressing the 'OK' button.

Attention:

The function returns immediately and is not blocking the calling application whilst establishing a connection to the GPS receiver. Check the result value before starting to receive data.
During a normal call the function returns immediately a '1' indicating that the driver is in pending state.
Your application should wait for a certain amount of time and call this function again to see, if the connection is still pending. A '0' as the return value indicates a proper established connection. Alternatively the function *BT_GetState* might be called to check wether the connection is established or not.

After the connection is established the driver is automatically receiving data from the GPS receiver. Use the functions *BT_ReadNMEAString* or *BT_ReadNMEAData* to read the data.

Once the connection is established, the AutoOFF Feature of the Handheld is disabled to avoid a automatic switch off after the preset time of inactivity through the user and a loss of the bluetooth connection. In this mode the Handheld is draining battery power by running the application and the Bluetooth conection. The AutoOFF feature is enabled again after a *BT_Disconnect* call or if the application is closed. For saving battery power it is not recommended to establish a connection automatically as soon as the application starts, but only when navigation data is required.

4.3 BT_Disconnect

Drop the current connection and enable the AutoOFF feature of the Handheld again.

Usage:

```
result = BT_Disconnect()
```

Returns:

See result list

Attention:
The function returns immediately and is not blocking the calling application whilst dropping a connection from the GPS receiver. Check the result value before proceeding, if required.
During a normal call the function returns immediately a '1' indicating that the driver is in pending state.
Your application should wait for a certain amount of time and call this function again to see, if the disconnection is still pending. A '0' as the return value indicates a proper disconnect. Alternatively the function *BT_GetState* might be called to check wether the connection has been dropped.

4.4 BT_GetDeviceAddress

After a successful discovery this function can be called to get the device address of the Bluetooth device which has been selected by the user. This address (string) can be stored in the application. Storing the address within your application is useful to connect in future sessions with *BT_SetDeviceAddress* without the need for a time consuming discovery process.

Usage:

```
DeviceAddressString = BT_GetDeviceAddress()
```

Returns:

String containing the current device address or an empty string.
Sample of an Address String:
00:C4:35:D5:BA

4.5 BT_SetDeviceAddress

Set the device address of the Bluetooth device to connect to. If the address of a Bluetooth device is already available from a former session (*BT_GetDeviceAddress*), there is no need to start a discovery process. *BT_Connect* can be called after this function immediately.

Usage:

```
result = BT_GetDeviceAddress(DeviceAddressString)
```

The *DeviceAddressString* (**"xx:xx:xx:xx")** has to be in the same format as it is returned by calling *BT_GetDeviceAddress*.

Returns:

See result list

4.6 BT_ReadNMEAString

Read the current GPS data string from the receive buffer. The data string which is returned contains a valid NMEA-0183 GPS data string, which can be used in the application. The checksum of the data is already verified by the driver **and not part of the result string**.

Usage:

```
data = BT_ReadNMEAString(selectString)
```

Returns:

NMEA GPS data string without checksum or empty string if no valid data is available

Sample:

```
Dim data

data = BT_ReadNMEAString("GGA")
```

After this call data contains the string:

```
"$GPGGA,134829.48,1126.6639,S,11133.3299,W,1,07,1.0,,,,,,"
```

If one of the follwing Strings are passed when calling, this function will only return the related Data strings once it has been received one from the GPS receiver:

- GGA
- RMC

- GSA
- GSV
- VTG
- GLL
- RMA
- RMB

See the NMEA-0183 chapter in this document for more details about the format of NMEA messages.

4.7 BT_ReadNMEAData

Read the current GPS data string in Memory buffer passed from the application. The data which is written in the buffer contains a valid NMEA-0183 GPS data message, which can be used in the application. The checksum of the GPS data has been verified already by the driver **and is not part of the data which is returned.**

Usage:

```
result = BT_ReadNMEAData(buffer, size, SelectString)
```

Returns:

See result list

If one of the follwing Strings are passed when calling, this function will only return the related Data strings once it has been received one from the GPS receiver:

- GGA
- RMC
- GSA
- GSV
- VTG
- GLL
- RMA
- RMB

See the NMEA-0183 chapter in this document for more details about the format of NMEA messages.

4.8 BT_FlushReceiveBuffer

Flush the receive buffer in the Bluetooth stack in case of error conditions or for initalizing purposes.

Usage:

```
result = BT_FlushReceiveBuffer()
```

Returns:

See result list

4.9 BT_ExtractParameter

Pass a GPS data string and the number of the paramter which should be extracted from the string. There is no need to write an own GPS message parser in the application.

Usage:

```
resultString = BT_ExtractParameter(NMEAString, number)
```

Returns:

Result string or empty string

Sample:

Assume that the follwing GPS string has been received:

```
NMEAData = "$GPGGA,134829.48,1126.6639,S,11133.3299,W,1,07,1.0,,,,,,"

ResultString = BT_ExtractParamter(NMEAData, 1)
MsgBox("Data: " & ResultString)
```

The MessageBox will display:

```
Data: 134829.48
```

You can continue to parse the data for extracting more parameters:

```
ResultString = BT_ExtractParamter(NMEAData, 3)
MsgBox("Data: " & ResultString)
```

The MessageBox will display:

```
Data: 3
```

4.10 BT_GetState

Returns the current state of the Bluetooth driver. The following states are defined

NotConnected	0
Pending	1
Connected	2
ConnectionFailed	3
ConnectionLost	4
NameRetrieval	5

The state 'Pending' is reported, if a connection is pending after calling *BT_Connect* or if a connection is closed using *BT_Disconnect.*

The state 'NameRetrieval' is reported, if the Bluetooth driver requests the Device Name from the GPS receiver after calling *BT_GetDeviceName*.

Usage:

```
state = BT_GetState()
```

4.11 BT_GetDeviceName

Get the Device Name of the Bluetooth Device, which has been selected during the discovery process or after setting the Bluetooth device address manually using BT_SetDeviceAddress.

Usage:

```
Name = BT_GetDeviceName()
```

4.12 BT_About

Show the extension's About Box.

Usage:

```
BT_About()
```

5. Result Values

The GPS Bluetooth driver returns the following return codes:

BT_NoError 0
BT_Pending 1
BT_ErrNoMemory 2
BT_ErrNoBtLib 3
BT_ErrNoBluetooth 4
BT_ErrOpenFailure 5
BT_ErrDiscoverFailure 6
BT_ErrIncorrectState..... 7
BT_ErrFailed 8
BT_ErrBusy..... 9
BT_ErrTooBig 10
BT_UserCancel..... 11
BT_NotConnected..... 12
BT_WrongParameter 13
BT_Timeout..... 14
BT_NullPointer..... 15

6. NMEA-0183 MESSAGES

NMEA 0183 is a standard protocol, used by GPS receivers to transmit data. NMEA 0183 sentences are all ASCII. Each sentence begins with a dollarsign ($) and ends with a carriage return linefeed (<CR><LF>). The GPS data string is comma delimited. Some GPS receivers do not send all sentences described below or do not include the full string content. Please have a look in the documentation of the specific GPS receiver or contact the manufacturer for details about supported sentences and their content. A checksum is optionally added (in a few cases it is mandatory). Following the $ is the address field aaccc. aa is the device id. GP is used to identify GPS data. Transmission of the device ID is usually optional. ccc is the sentence formatter, otherwise known as the sentence name.

Here are some of the most important sentences:

- **GPGGA**
- **GPRMC**
- **GPGSA**
- **GPGSV**
- **GPVTG**
- **GPGLL**

6.1 GGA – Global Positioning System Fix Data

Time, position and fix related data for a GPS receiver.

$GPGGA,hhmmss.dd,xxmm.dddd,<N|S>,yyymm.dddd,<E|W>,v,ss,d.d,h.h,M,g.g,M, a.a,xxxx*hh

1	hhmmss.dd	UTC time hh = hours mm = minutes ss = seconds dd = decimal part of seconds
2	xxmm.dddd	Latitude xx = degrees mm = minutes dddd = decimal part of minutes
3	<N|S>	Either character N or character S, (N = North, S = South)
4	yyymm.dddd	Longitude yyy = degrees mm = minutes dddd = decimal part of minutes
5	<E|W>	Either character E or character W, E = East, W = West
6	v	GPS Quality indicator/Fix valid indicator 0 = GPS Fix not valid 1 = GPS Fix valid 2 = Diff. GPS fix valid
7	ss	Number of satellites used in position fix, 00-12. Fixed length
8	d.d	HDOP – Horizontal dilution of position
9	h.h	Antenna altitude (above/below mean-sea-level, geoid)
10	M	Meters (Antenna height unit)
11	g.g	Geoidal separation (Diff. between WGS-84 earth ellipsoid and mean-sea-level. '-' is geoid is below WGS-84 ellipsoid
12	M	Meters (Units of geoidal separation)
13	a.a	Age in seconds since last update from diff. reference station
14	xxxx	Diff. reference station ID#
15	*hh	Checksum

Example:

```
$GPGGA,134829.48,1126.6639,S,11133.3299,W,1,07,1.0,,,,,,*15
```

6.2 RMC – Recommended Minimum Specific GNSS Data

Time, date, position, course and speed data.

$GPRMC,hhmmss.dd,S,xxmm.dddd,<N|S>,yyymm.dddd,<E|W>,s.s,h.h,ddmmyy,d.d, <E|W>,M*hh

1	hhmmss.dd	UTC time hh = hours mm = minutes ss = seconds dd = decimal part of seconds
2	S	Data status indicator A valid V invalid
3	xxmm.dddd	Latitude xx = degrees mm = minutes dddd = decimal part of minutes
4	<N|S>	Either character N or character S, (N = North, S = South)
5	yyymm.dddd	Longitude yyy = degrees mm = minutes dddd = decimal part of minutes
6	<E|W>	Either character E or character W, E = East, W = West
7	s.s	Speed over ground in knots
8	h.h	Heading – Track made good in degrees True
9	ddmmyy	UT Date dd = day mm = month yy = year
10	d.d	Magnetic variation degrees (Easterly var. subtracts from true course)
11	<E|W>	Declination. Either character E or character W, E = East, W = West
12	M	Mode indicator A = autonomous N = data not valid
13	*hh	Checksum

Example:
```
$GPRMC,134829.486,A,1126.6639,S,11133.3299,W,58.31,309.62,110200,,
,A*14
```

6.3 GSA – DOP and Active Satellites

GPS receiver operating mode, satellites used in the navigation solution reported by the GGA sentence, and DOP values.

$GPGSA,a,b,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,p.p,h.h,v.v*hh

1	a	Mode: M = Manual, forced to operate in 2D or 3D mode. A = Automatic, allowed to automatically switch 2D/3D.
2	b	Mode: 1 = Fix not available 2 = 2D 3 = 3D
3	xx	ID (PRN) numbers of GPS satellites used in solution
4	p.p	PDOP
5	h.h	HDOP
6	v.v	VDOP
7	*hh	Checksum

Example:
```
$GPGSA,A,3,03,15,17,18,22,23,,,,,,,4.7,3.7,2.9*37
```

6.4 GSV – Satellites in view

Number of satellites in view, satellite ID (PRN) numbers, elevation, azimuth, and SNR value. The information for four satellites maximum per one message, additional messages up to maximum of eight sent as needed. The satellites are in PRN number order. Before a position fix is acquired the information contains only the SNR (signal to noise ratio) value. After a fix is acquired, also the elevation and azimuth angles are added. Note that there can be also "theoretical" satellites in the GSV message. These are satellites of which the angles (elevation, azimuth) are known but for some reason, e.g. due to an obstruction, have not been found by the receiver. The SNR value for these satellites is therefore zero.

$GPGSV,n,m,ss,xx,ee,aaa,cn,…… ,xx,ee,aaa,cn*hh

1	n	Total number of messages, 1 to 9
2	m	Message number, 1 to 9
3	ss	Total number of satellites in view
4	xx	Satellite ID (PRN) number
5	ee	Satellite elevation, degrees 90 max
6	aaa	Satellite azimuth, degrees True, 000 to 359
7	cn	SNR (C/No) 00-99 dB-Hz. zero when not tracking
8	*hh	Checksum

Example:

```
$GPGSV,4,1,14,03,66,207,50,08,09,322,44,11,01,266,42,14,00,155,00*79
$GPGSV,4,2,14,15,41,088,48,17,21,083,44,18,57,087,51,21,57,173,50*78
$GPGSV,4,3,14,22,05,203,00,23,52,074,49,26,17,028,44,27,00,300,00*79
$GPGSV,4,4,14,28,32,243,00,31,48,286,00*70
```

6.5 VTG – Course Over Ground and Ground Speed

Course and speed

$GPVTG,h.h,T,m.m,M,s.s,N,s.s,K,M*hh

1	h.h	Heading
2	T	Degrees (heading units) Fixed 'T' indicates that track made good is relative to true north
3	m.m	Magnetic heading. Currently NULL (missing)
4	M	Degrees. Magnetic heading units. Currently NULL (missing)
5	s.s	Speed in knots
6	N	N indicating Knots (Speed unit)
7	s.s	Speed in km/h
8	K	K indicating km/h (Speed units)
9	M	Mode indicator: A = Autonomous

		N = Data invalid
10	*hh	Checksum

Example:
```
$GPVTG,202.60,T,,,0.38,N,0.7,K,A*0D
```

6.6 GLL – Geographic Position – Latitude/Longitude

Latitude and Longitude, UTC time of fix and status.

$GPGLL,xxmm.dddd,<N|S>, yyymm.dddd,<E|W>,hhmmss.dd,S,M*hh

1	xxmm.dddd	Latitude xx = degrees mm = minutes dddd = decimal part of minutes
2	<N|S>	Either character N or character S, (N = North, S = South)
3	yyymm.dddd	Longitude yyy = degrees mm = minutes dddd = decimal part of minutes
4	<E|W>	Either character E or character W, E = East, W = West
5	hhmmss.dd	UTC time hh = hours mm = minutes ss = seconds dd = decimal part of seconds
6	S	Data status indicator A valid V invalid
7	M	Mode indicator A = autonomous N = data not valid
8	*hh	Checksum

Example:
```
$GPGLL,6016.3073,N,02458.3791,E,134157.48,A,A*26
```