handheld competence
Jochen Hammer

palm OS CERTIFIED DEVELOPER

RFCOMM
Bluetooth Driver

Mobile AppDesigner Extension
for
Palm OS©

Disclaimer

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of Handheld Competence.

Handheld Competence and the Handheld Competence Logo, are trademarks of Handheld Competence. The Bluetooth Logo may be registered in some jurisdictions. Palm is a trademark of Palmsource, Inc. Palm OS is a registered trademark of Palmsource, Inc. All other company and product names and logos may be trademarks of their respective owners.

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	2005-03-11	Initial release
V1.1	2007-04-28	Small modification to work on Janam XP30 devices

1. Abstract

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

This driver offers all the required functionality for RFCOMM purposes:

- Initalise Bluetooth connection in either Master or Client mode
- Discover Bluetooth Devices from within your application
- Connect to a selected RFCOMM device
- Set and Get the address of the selected Bluetooth device
- Retrieve the name of the Bluetooth device once a connection is established
- Check the status of the Bluetooth connection
- Get number of received bytes
- Send data as string or a raw data stream by passing a buffer
- Receive data as string or a raw data stream by passing a buffer
- Disconnect

This 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.

2. Content

3. Introduction

The RFCOMM 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 sending the data interrupt driven, so the User can continue using the application without any delays whilst sending or receiving data.

Once a Bluetooth connection to a selected device is established data can be send or received with a single function call.

The driver offers all functionality required for running the RFCOMM connection in either client or server mode.

In difference to a hard wired serial connection, a Bluetooth connection is based upon a server/client role architecture. One Bluetooth device acts as the server and offers its Bluetooth services, the other devices acts as the client and connects to the server. After a discovery process, the client requests all services from a server (SDP: service discovery process) .

In most cases the Palm will act as the Bluetooth client, which requests a RFCOMM connection from the server. For instance, a GPS receiver acts as a Bluetooth server and the client can connect to the GPS device after the discovery process. During the service discovery process, the Palm client gets all information about the available services from the server. This could be the baud rate the data is transmitted, for instance. Once this information has been received from the client, it can connect to the server and receive the data, which is transmitted.

Other Bluetooth devices, which offer RFCOMM services as Bluetooth server:

- Cell phones
- Printer with Bluetooth interface
- Barcode scanner
- USB Dongles
- Digital cameras with Bluetooth interface
- …

If another Bluetooth equipped device, e.g. a PDA or a notebook, should connect to the Palm to exchange data, then the Palm acts as the Bluetooth server and offers a RFCOMM service to all listening devices. For instance, if you like to write a chat application for two Palm PDAs, one Palm has to act as server whereas the other acts as the client.

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

4. API Reference

The RFCOMM driver supports the following 17 functions, which are described in detail below:

1. BT_Discover
2. BT_Connect
3. BT_Disconnect
4. BT_GetDeviceAddress
5. BT_SetDeviceAddress
6. BT_ReceiveDataStr
7. BT_ReceiveDataBuf
8. BT_FlushReceiveBuffer
9. BT_SendString
10. BT_SendData
11. BT_GetState
12. BT_GetDeviceName
13. BT_GetBytesAvailInt
14. BT_GetBytesAvailStr
15. BT_InitAsServer
16. BT_GetPacketSize
17. 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()
```

Client/Server Mode:
Available in Client mode only.

Returns:

See result list

4.2 BT_Connect

Connect to the Bluetooth device 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, Blocking)
```

Client/Server Mode:
Available in Client and Server mode. When the driver is initialised in server mode, the second parameter 'Blocking' is ignored.

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:
"Bluetooth connection established."
"Bluetooth connection failed."
"Bluetooth connection lost."
"Bluetooth connection dropped."
These Messages boxes have to be confirmed from the user by pressing the 'OK' button.

Passing the boolean value '*Blocking'* results in (**ONLY IN CLIENT MODE**):

False Returns immediately and do not block the calling application until the connection to the Bluetooth device is established. Check the result value before starting to send 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 whether the connection is established or not. (**DEFAULT**)

True Passing true will result in a function call, which is blocking the calling application until the connection to the Bluetooth device is established or a timeout period of 8 seconds exceeds.

Attention:

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 connection. 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 RFCOMM functionality is required.

4.3 BT_Disconnect

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

Usage:

```
result = BT_Disconnect(Blocking)
```

Client/Server Mode:
Available in Client and Server mode. When the driver is initialised in server mode, the parameter 'Blocking' is ignored.

Returns:

See result list

Passing the boolean value '*Blocking'* results in (**ONLY IN CLIENT MODE**):

False Returns immediately and do not block the calling application until the connection to the Bluetooth device is dropped. 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 whether the connection is dropped or not. (**DEFAULT**)

True Passing true will result in a function call, which is blocking the calling application until the disconnection to the Bluetooth device is done or a timeout period of 8 seconds exceeds.

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()
```

Client/Server Mode:
Available in Client mode only.

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*.

Client/Server Mode:
Available in Client mode only.

Returns:

See result list

4.6 BT_ReceiveDataStr

Read the data, which has been received and return it as string. The function *BT_GetBytesAvailInt* should be called, to check if data is available in the receive buffer.

Usage:

```
String = BT_ReceiveDataStr()
```

Client/Server Mode:
Available in Client and Server mode.

Returns:

The string “Error” is returned, if an error has occurred.

4.7 BT_ReceiveDataBuf

Read the data, which has been received in a memory buffer, which is passed from the application. The function *BT_GetBytesAvailInt* should be called before, to check if data is available in the receive buffer.

Usage:

```
result = BT_ReceiveDataBuf(buffer, size)
```

Client/Server Mode:
Available in Client and Server mode.

Returns:

See result list

Example:

```
Dim buffer,error, size

‘ Check for any received data
size = BT_GetBytesAvailInt()

If size = 0 Then
     ‘ No data in receive buffer ==> leave
     Exit
Endif

'Create a buffer, which is slightly larger
buffer = MemoryAllocate(size + 10)

If buffer = 0 Then
     MsgBox("Error creating buffer")
```

```
     Exit
Endif

'Initialise the buffer with 0
buffer = MemorySet(buffer, 0, size+10)

If buffer = 0 Then
     MsgBox("Error initialising buffer")
     MemoryFree(buffer)
     Exit
Endif

result = BT_ReceiveDataBuf(buffer, size+10)

If result = 0 Then
   ' OK, we have the received data in our buffer now….
   ' …
Else
   ' Error handling…
Endif

MemoryFree(buffer)
```

4.8 BT_FlushReceiveBuffer

Flush the receive buffer.

Usage:

```
result = BT_FlushReceiveBuffer()
```

Client/Server Mode:
Available in Client and Server mode.

Returns:

See result list

4.9 BT_SendString

Send the string data, which is passed as first parameter. The second parameter specifies, if the sending shall be done in blocking or non-blocking mode.

Usage:

```
result = BT_SendString(String, Blocking)
```

Client/Server Mode:
Available in Client and Server mode. When the driver is initialised in server mode, the parameter 'Blocking' is ignored.

Passing the boolean value '*Blocking'* results in **(ONLY IN CLIENT MODE):**

False Returns immediately and do not block the calling application until the data is send. Check the result value before proceeding, if required.
During a call the function returns immediately a '1' indicating that the driver is in pending or sending state. Your application should wait for a certain amount of time and call this function again to see, if the sending is still pending. A '2' as the return value indicates that the data has been send. Alternatively the function *BT_GetState* might be called to check whether the sending is still in progress. (**DEFAULT**)

True Passing true will result in a function call, which is blocking the calling application until the sending is done or an error occurs.

Returns:

See result list

Attention:

In client mode the maximum length of the string which can be send is limited by the RFCOMM buffer in the receiving device. You can receive this value by calling *BT_GetPacketSize* once a connection has been established. If you send a string which is longer, then the error code '10' (ErrTooBig) is returned. If you want to send a longer string, you have to split it in smaller packets.

Sample:

Send some sample strings.
For simplicity we assume that a connection is established in client mode and we do not verify the state before and after sending.

```
Dim result, string
Dim length, maxpacketSize
Dim temp, i
Dim packetnumber, rest
```

```
…
string = "Bluetooth Device"

…
'Send a string variable non-blocking
result = BT_SendString(string,false)

…
'Send some static text non-blocking
result = BT_SendString("RFCOMM Test", false)


…
'Send a really long string
string = "… a really long string is stored here …"

'Obtain maximum packet size for the RFCOMM connection and
'get the length of the string

maxpacketSize = BT_GetPacketSize()
'Limit the packet to 50 bytes (just as an example)
If maxpacketSize > 50 Then
     maxpacketSize = 50
Endif

length = Len(string)

If length <= maxpacketSize Then

     'String can be send with one non-blocking function call
     'completely
     result = BT_SendString(string,false)

Else
     'String is longer and has to be splitted in smaller
     'packets; block the application until the device has
     'received each packet

     packetnumber = Int(length/maxpacketSize)

     ' loop for sending packets
     for i = 0 to packetnumber-1
          temp = Mid(string, i*maxpacketSize, maxpacketSize)
          result = BT_SendString(temp, true)
     next i

     rest = length - (packetnumber * maxpacketSize)
```

```
      If rest > 0 Then
         temp = Mid(string, i*maxpacketSize, rest)
         result = BT_SendString(temp, false)
      Else
         result = BT_SendString(" ", false)
      Endif
Endif
```

4.10 BT_SendData

Send data to the connected device. Pass a pointer on the buffer containing the data, which shall be send, pass the start position within the buffer and the length. The last parameter specifies, if the sending shall be done in blocking or non-blocking mode.

Usage:

```
result = BT_SendData(buffer, start, length, blocking)
```

Client/Server Mode:
Available in Client and Server mode. When the driver is initialised in server mode, the parameter 'Blocking' is ignored.

Returns:

See result list

Passing the boolean value '*Blocking'* results in (**ONLY IN CLIENT MODE**):

False Returns immediately and do not block the calling application until the data is send. Check the result value before proceeding, if required.
During a call the function returns immediately a '1' indicating that the driver is in pending or sending state. Your application should wait for a certain amount of time and call this function again to see, if the sending is still pending. A '2' as the return value indicates that the data has been send. Alternatively the function *BT_GetState* might be called to check whether the sending is still in progress. (**DEFAULT**)

True Passing true will result in a function call, which is blocking the calling application until the data is send or an error occurs.

Attention:

In client mode the maximum length of the string which can be send is limited by the RFCOMM buffer in the receiving device. You can receive this value by calling *BT_GetPacketSize* once a connection has been established. If you send a string which is longer, then the error code '10' (ErrTooBig) is returned. If you want to send a longer string, you have to split it in smaller packets.

Example:

The following example sends a buffer of 5 byte

Note: The Intellisync Memory extension has to be included in your project.

```
Dim buffer,error

'Create a small buffer
buffer = MemoryAllocate(5)

If buffer = 0 Then
      MsgBox("Error creating buffer")
      Exit
Endif

'Initialise the buffer with 0's
buffer = MemorySet(buffer, 0, 5)

If buffer = 0 Then
      MsgBox("Error initialising buffer")
      MemoryFree(buffer)
      Exit
Endif

'Write some data in our buffer
MemorySetByte(buf,0,&H41) 'A
MemorySetByte(buf,1,&H42) 'B
MemorySetByte(buf,2,&H43) 'C
MemorySetByte(buf,3,&H44) 'D
MemorySetByte(buf,4,&H45) 'E

'Finally send this buffer in non-blocking mode
'For simplicity we assume that a connection is established
'and we do not verify the state here…

…

error = BT_SendData(buffer,0,5,false)

…

'Free buffer
MemoryFree(buffer)
```

4.11 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
Sending	6

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 Bluetooth device after calling *BT_GetDeviceName*.

Usage:

```
state = BT_GetState()
```

Client/Server Mode:
Available in Client and Server Mode

4.12 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()
```

Client/Server Mode:
Available in Client mode only.

4.13 BT_GetBytesAvailInt

Read the number of bytes in the receive buffer and return this value as a integer.

Usage:

```
numBytes = BT_GetBytesAvailInt()
```

Client/Server Mode:
Available in Client and Server Mode

4.14 BT_GetBytesAvailStr

Read the number of bytes in the receive buffer and return this value as a string.

Usage:

```
numBytesString = BT_GetBytesAvailStr()
```

Client/Server Mode:
Available in Client and Server Mode

4.15 BT_InitAsServer

Initialise the Palm to act as Bluetooth Server or Client. Pass 'TRUE' to init as Server and 'FALSE' as Client (**Default**).

Usage:

```
error = BT_InitAsServer(True)
```

Client/Server Mode:
This function should be called during the application start.

Returns:

See result list

4.16 BT_GetPacketSize

Get the maximum packet size of the RFCOMM connection once a connection has been established. This value indicates the maximum number of bytes the RFCOMM connection of the device can handle at a time. If a buffer or a string which exceeds this size is send the error *ErrTooBig* is returned after calling the command *BT_SendString* or *BT_SendData*.

Usage:

```
size = BT_GetPacketSize()
```

Client/Server Mode:
Available in Client Mode only.

4.17 BT_About

Show the extension's About Box.

Usage:

```
BT_About()
```

5. Result Values

The Bluetooth driver returns the following return codes:

Result:	Description:	
0	**NoError**	OK
1	**Pending**	The Bluetooth radio is busy with connecting, disconnecting, or sending data
2	**ErrNoMemory**	The driver was not able to allocate enough memory (severe error)
3	**ErrNoBtLib**	No Bluetooth library found
4	**ErrNoBluetooth**	No Bluetooth radio available
5	**ErrOpenFailure**	Error opening the Bluetooth library
6	**ErrDiscoverFailure**	Error discovering Bluetooth devices
7	**ErrIncorrectState**	The driver is in a wrong state, e.g not connected whilst trying to send data
8	**ErrFailed**	An function call failed (severe error)
9	**ErrBusy**	The driver is busy
10	**ErrTooBig**	The data which has been send exceeds the maximum packet size of the printer
11	**UserCancel**	The User cancelled the discovery process
12	**NotConnected**	The connection is not established
13	**WrongParameter**	A wrong parameter has been passed

14	**NullPointer**	A NULL pointer has been passed (severe error)
15	**InterfaceNotOpen**	SERVER Mode only; the RFCOMM interface is not open
16	**DiscoverCancelled**	The user cancelled the discovery process