Using the internal GPS receiver of a (Nokia) Symbian mobile phone

The following has been tested successfully with a Nokia N82.

Usually it is not possible to transfer GPS data from an internal GPS receiver of a mobile phone to external devices via bluetooth but I have found this java based tool for Symbian phones:
http://www.symarctic.com/beta/static.php?page=extgps_download

Download ExtGPS with your mobile phone, install it and start it.
Now the following steps have to be taken (as root):

```
sdptool search SP
```

The output looks like this:

```
Inquiring ...
Searching for SP on XX:XX:XX:XX:XX:XX ...
Service Name: Data Transfer
Service Description: Version
Service RecHandle: 0x10019
Service Class ID List:
  "Serial Port" (0x1101)
Protocol Descriptor List:
  "L2CAP" (0x0100)

Service Name: Symarctic ExtGPS
Service Description: Share phone's built-in GPS module via Bluetooth
Service Provider: Symarctic Solutions
Service RecHandle: 0x1001f
Service Class ID List:
  "Serial Port" (0x1101)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 5
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
```

The important parts are:

```
Searching for SP on XX:XX:XX:XX:XX:XX
Channel: 5
```

“XX“ has to be replaced with your MAC adress.

Now continue with the creation of the interface for ExtGPS in Linux. "0" stands for the number of the interface, followed by the MAC adress and the channel number.

```
rfcomm bind 0 XX:XX:XX:XX:XX:XX 5
```

After this is done you can test your GPS connection:

```
cat /dev/rfcomm0
```

You should see a lot of GPS data passing by. You can stop the readout with Ctrl+C.

Now edit the file /etc/bluetooth/rfcomm.conf as follows:

```
#
# RFCOMM configuration file.
#

rfcomm0 {
#     # Automatically bind the device at startup
      bind yes;
#
#     # Bluetooth address of the device
      device XX:XX:XX:XX:XX:XX;
#
#     # RFCOMM channel for the connection
      channel     5;
#
#     # Description of the connection
      comment "N82 GPS";
#}
```

First start ExtGPS on the mobile phone then start your computer. Normally the device /dev/rfcomm0 should be present. This didn't work for me so I added the following line to /etc/rc.d/rc.local:

```
rfcomm bind 0 XX:XX:XX:XX:XX:XX 5
```

Now /dev/rfcomm0 should start and connect automatically.

Using a „normal“ Bluetooth GPS receiver

Turn your GPS receiver on. On the computer start the scanning process:

```
hcitool scan
Scanning ...
      XX:XX:XX:XX:XX:XX NOKIA N82
```

Because I don't own a bluetooth GPS receiver I have to continue using my mobile phone's receiver.
We will now search for the serial port:

```
sdptool browse
Inquiring ...
Browsing XX:XX:XX:XX:XX:XX ...
Service Name: Symarctic ExtGPS
Service Description: Share phone's built-in GPS module via Bluetooth
Service Provider: Symarctic Solutions
Service RecHandle: 0x10011
Service Class ID List:
  "Serial Port" (0x1101)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
```

```
    Channel: 5
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
```

The important parts are:

```
Searching for SP on XX:XX:XX:XX:XX:XX
Channel: 5
```

“XX“ has to be replaced with your MAC adress.

Now create the interface for ExtGPS in Linux. "0" stands for the number of the interface, followed by the MAC adress and the channel number.

```
rfcomm bind 0 XX:XX:XX:XX:XX:XX 5
```

Now you can test the GPS connection:

```
cat /dev/rfcomm0
```

You should see a lot of GPS data passing by. You can stop the readout with Ctrl+C.

When this is done edit the file /etc/bluetooth/rfcomm.conf as follows:

```
#
# RFCOMM configuration file.
#

rfcomm0 {
#     # Automatically bind the device at startup
      bind yes;
#
#     # Bluetooth address of the device
      device XX:XX:XX:XX:XX:XX;
#
#     # RFCOMM channel for the connection
      channel     5;
#
#     # Description of the connection
      comment "N82 GPS";
#}
```

Start your bluetooth GPS receiver before you start your computer is started. The device /dev/rfcomm0 should be present now. This didn't work for me so I added the following to /etc/rc.d/rc.local:

```
rfcomm bind 0 XX:XX:XX:XX:XX:XX 5
```

Now /dev/rfcomm0 should start and connect automatically.