

HE Windows CE 6.0 User Guide

1VV0300927 rev. 2 - 2012-03-08



Disclaimer

The information contained in this document is the proprietary information of Telit Communications S.p.A. and its affiliates ("TELIT"). The contents are confidential and any disclosure to persons other than the officers, employees, agents or subcontractors of the owner or licensee of this document, without the prior written consent of Telit, is strictly prohibited.

Telit makes every effort to ensure the quality of the information it makes available. Notwithstanding the foregoing, Telit does not make any warranty as to the information contained herein, and does not accept any liability for any injury, loss or damage of any kind incurred by use of or reliance upon the information.

Telit disclaims any and all responsibility for the application of the devices characterized in this document, and notes that the application of the device must comply with the safety standards of the applicable country, and where applicable, with the relevant wiring rules.

Telit reserves the right to make modifications, additions and deletions to this document due to typographical errors, inaccurate information, or improvements to programs and/or equipment at any time and without notice. Such changes will, nevertheless be incorporated into new editions of this application note.

Copyright: Transmittal, reproduction, dissemination and/or editing of this document as well as utilization of its contents and communication thereof to others without express authorization are prohibited. Offenders will be held liable for payment of damages. All rights are reserved.

Copyright © Telit Communications SpA 2011.



Applicable Products

Product
HE863-EUx
HE910



Contents

1. Introduction	5
1.1. Scope	5
1.2. Audience	5
1.3. Contact Information, Support	5
1.4. Product Overview	5
1.5. Document Organization	6
1.6. Text Conventions	6
1.7. Document history	6
2. System setup	7
2.1. General Overview	8
2.2. Pre-requirements	10
2.3. Adding HE support in the OS image using the binary	12
3. Using HE modules	14
3.1. The serial API	14
3.1.1. CreateFile	14
3.1.2. WriteFile	16
3.1.3. ReadFile	17
3.1.4. CloseHandle	19
3.2. PPP Connection	20
3.3. WinCE Limitations	21



1. Introduction

1.1. Scope

This user guide serves the following purpose:

- Provides details about the HE Telit Modules.
- Describes how to compile, include in a kernel image and use the HE Windows CE 6.0 driver for a headless system.

1.2. Audience

This User Guide is intended for software developers who develop applications using HE modem.

1.3. Contact Information, Support

For general contact, technical support, to report documentation errors and to order manuals, contact Telit's Technical Support Center (TTSC) at:

TS-EMEA@telit.com
TS-NORTHAMERICA@telit.com
TS-LATINAMERICA@telit.com
TS-APAC@telit.com

Alternatively, use:

<http://www.telit.com/en/products/technical-support-center/contact.php>

For detailed information about where you can buy the Telit modules or for recommendations on accessories and components visit:

<http://www.telit.com>

To register for product news and announcements or for product questions contact Telit Technical Support Center (TTSC).

Our aim is to make this guide as helpful as possible. Keep us informed of your comments and suggestions for improvements.

Telit appreciates feedback from the users of our information.

1.4. Product Overview

HE modules contain a fully featured HSPA modem and UMTS/GSM/GPRS/EDGE module.



1.5. Document Organization

This manual contains the following chapters:

- "Chapter 1, Introduction" provides a scope for this manual, target audience, technical contact information, and text conventions.
- "Chapter 2, System setup" describes how to add the HE USB driver in a Windows CE 6.0 image.
- "Chapter 3, Device Driver" describes how to use the HE driver for interacting with the HE modem.

1.6. Text Conventions

This section lists the paragraph and font styles used for the various types of information presented in this user guide.



Danger - This information MUST be followed or catastrophic equipment failure or bodily injury may occur.



Caution or Warning - Alerts the user to important points about integrating the module, if these points are not followed, the module and end user equipment may fail or malfunction.



Tip or Information - Provides advice and suggestions that may be useful when integrating the module.

Format	Content
Arial monospaced	Windows CE shell commands at command prompt, filesystem paths, source code examples and menu items

All dates are in ISO 8601 format, i.e. YYYY-MM-DD.

1.7. Document history

Revision	Date	Changes
ISSUE #0	2011-04-15	First Release
ISSUE #1	2011-09-05	Added HE910 Support
ISSUE #2	2012-03-08	Added Windows CE Limitations paragraph



2. System setup

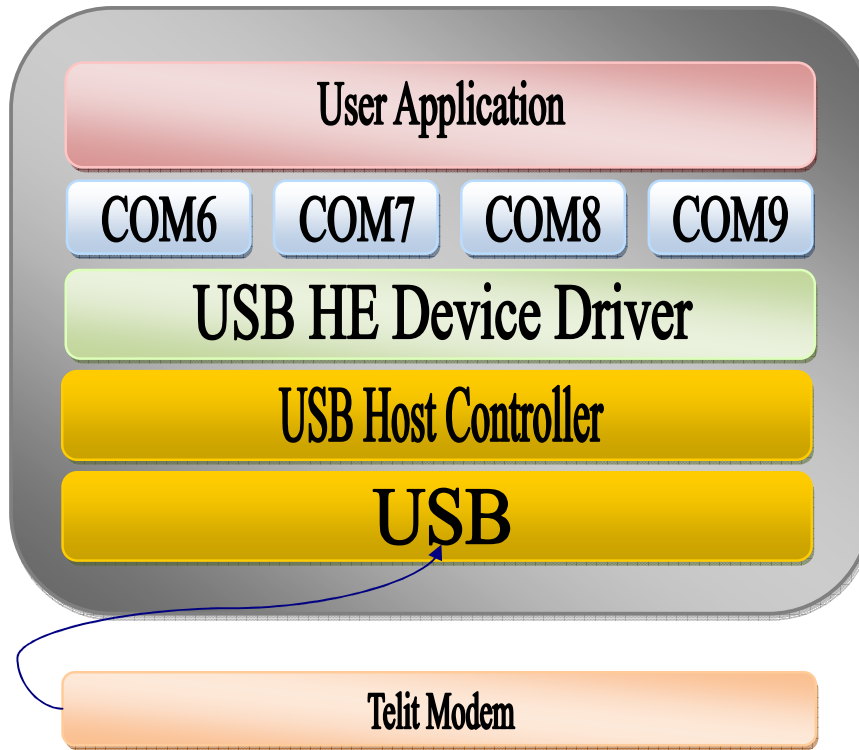
In this chapter it is described the general architecture of a Windows CE system with attached a HE module, the requirements needed for using the HE and how to integrate the HE device driver in a Windows CE image. The following table shows the reference platforms which this guide has been tested for:

Device	Operating System
AT92SAM9263-EK	Windows CE 6.0 R3



2.1. General Overview

In the following image it is shown a diagram of USB software/hardware interaction in a Windows CE system with a HE attached:



From the bottom of the stack:

- Telit modem connected through an USB port.
- USB: serial bus for interfacing devices to a host computer.
- USB Host Controller: a combination of hardware and software that is responsible for the following actions:
 - o Detecting the insertion and removal of USB devices
 - o Managing flow control between the host and USB devices
 - o Managing data flow between the host and USB devices
 - o Collecting status
 - o Providing power to attached USB devices



- USB HE Device Driver: piece of software that allows the HE to be seen by the user application as connected through serial ports rather than USB port.
- Virtual serial ports (COM6, COM7, COM8 and COM9): serial ports created by the HE device driver for accessing the modem; they can be used as real physical serial ports.

HE863:

- o COM6: Telit HSDPA modem port, used for normal modem/application interaction (e.g. AT commands sending, data connection...).
- o COM7: Telit on COM7.
- o COM8: Telit on COM8.
- o COM9: Telit Auxiliary

HE910:

- o COM6: Telit on COM7.
- o COM7: Telit on COM7.
- o COM8: Telit on COM8.
- o COM9: Telit Auxiliary

- User Application: piece of software written by the customer that uses HE features through virtual serial ports.

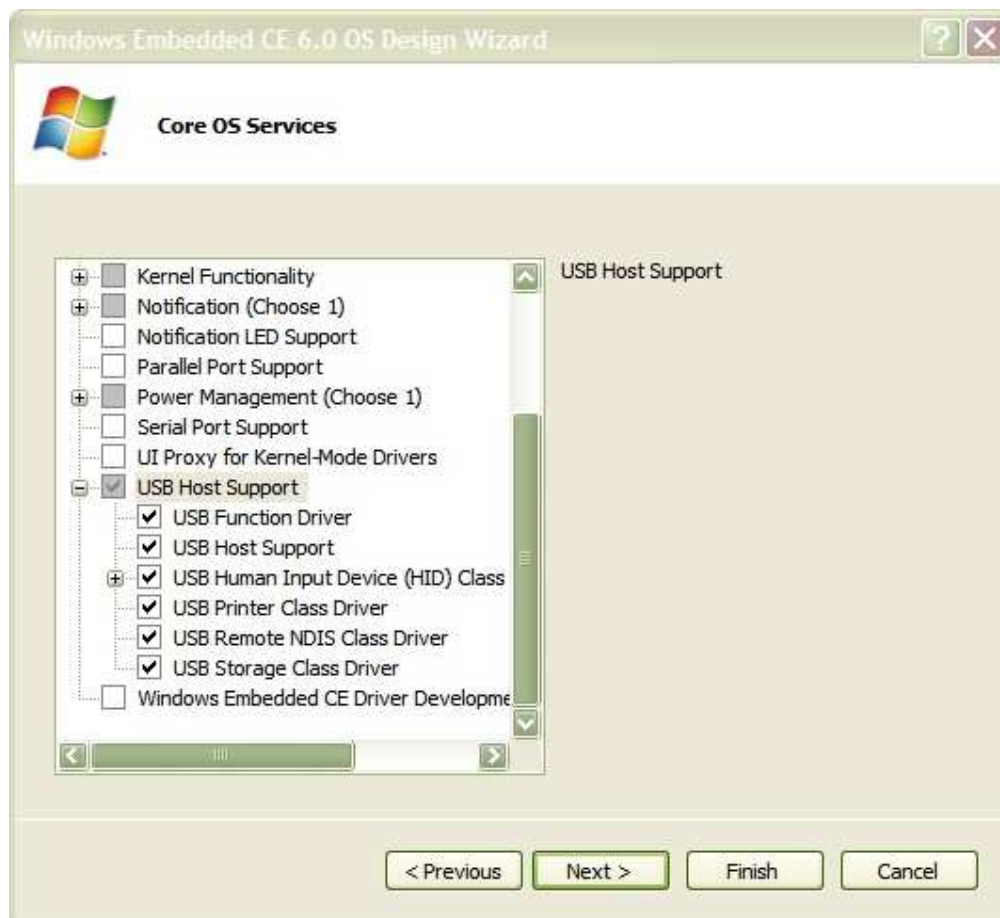


2.2. Pre-requirements

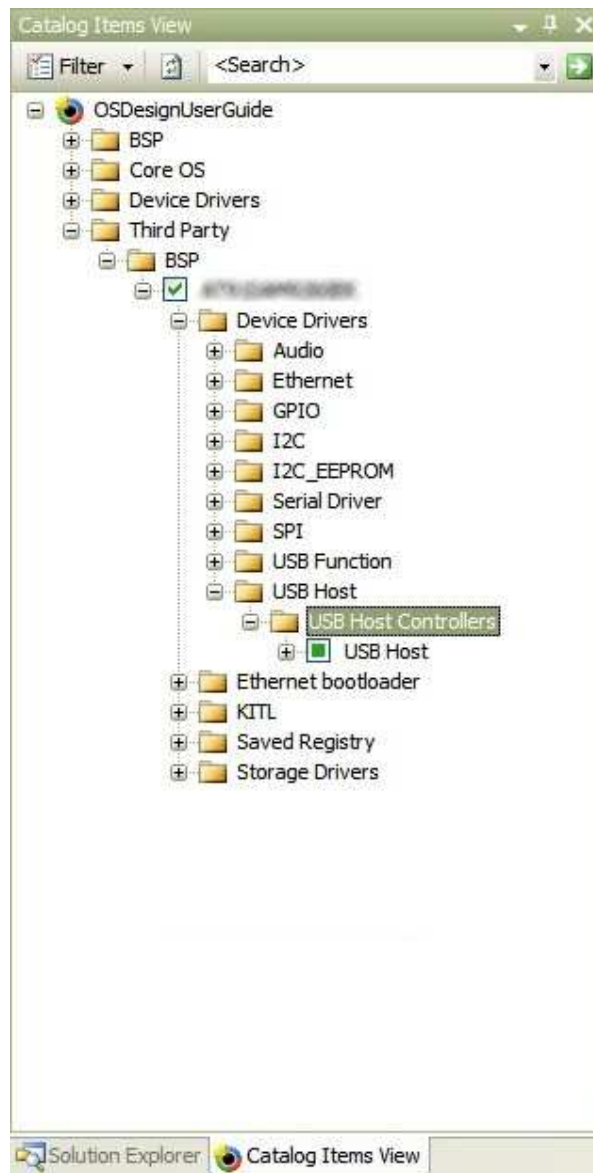
For correctly building a Windows CE 6.0 image with the HE USB driver on a headless system the following pre-requirements need to be satisfied:

- Microsoft Visual Studio 2005, with Platform Builder for CE 6.0.
- An OS design with the USB host controller supported.
- Knowledge of Windows CE OS image and subproject creation.

Please note that there are different ways for enabling the USB host controller and they depend on the considered hardware. In our test platform the USB host controller can be enabled during the initial image creation wizard (in the Core OS Services step), as the following image suggests:



or it can be added using the catalog as the following image suggests:



Please note again that this step could be different for your platform.



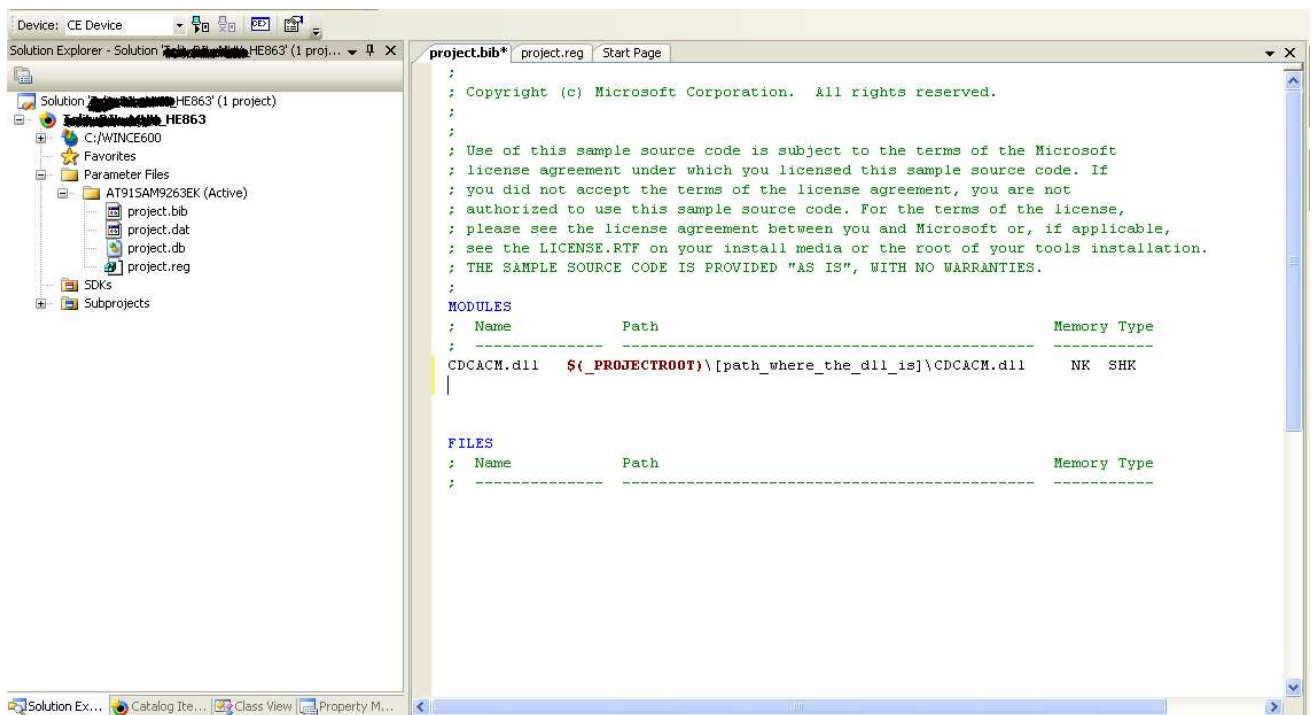
2.3. Adding HE support in the OS image using the binary

For adding HE support in the OS image, having the binary `CDCACM.dll`, follow these steps:

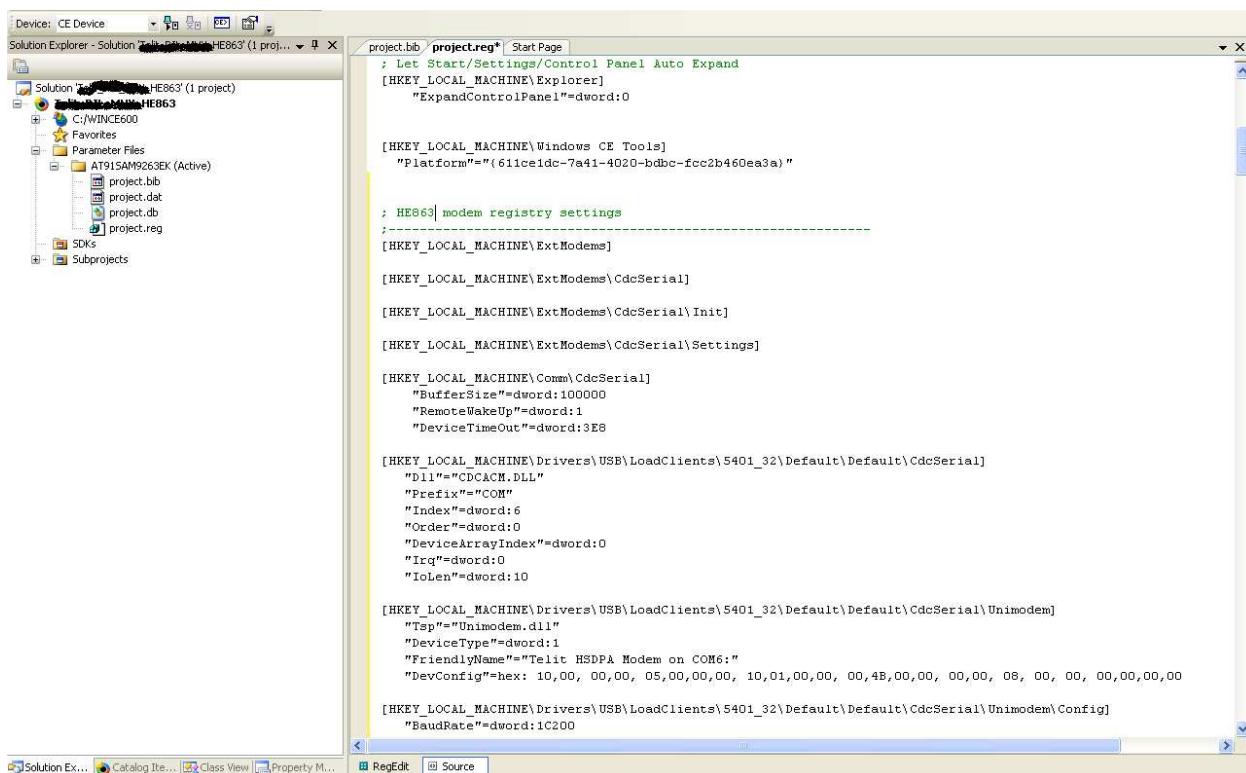
- In the *Solution Explore*, under the folder *Parameter Files* open the folders called as your BSP name and open the file *project.bib*. Under the *Modules* section add the following line:

```
CDCACM.dll  
$(_PROJECTROOT)\[path_where_the_dll_is]\CDCACM.dll      NK  
SHK
```

paying attention to change the path to the correct one (the place where the dll can be found).



- Open the file *project.reg* and, at its end, paste the content of the file *CDCACMDriver.reg*.



- Recreate the OS image.

Now you should have the HE driver integrated into your OS image. Start your system with the just created OS image and, after the boot, plug the device into your target: if the connection succeeds you should see the four virtual serial ports described in paragraph 2.1 ready to be used.



3. Using HE modules

In this chapter it is explained how to programmatically use HE modules (via the serial port API) and how to setup a PPP connection.

3.1. The serial API

Application can interact with the HE through the virtual serial ports created by the driver (for example COM6). Windows CE 6.0 has a complete API for dealing with serial ports; following you can find all the most important calls with code examples. Further information can be found in Microsoft Developer Network.

3.1.1. CreateFile

This function creates, opens, or truncates a file, COM port, device, service, or console. It returns a handle to access the object.

Header:

winbase.h

Library:

coredll.lib

Syntax:

```
HANDLE CreateFile(
    LPCTSTR lpFileName,
    DWORD dwDesiredAccess,
    DWORD dwShareMode,
    LPSECURITY_ATTRIBUTES lpSecurityAttributes,
    DWORD dwCreationDisposition,
    DWORD dwFlagsAndAttributes,
    HANDLE hTemplateFile
);
```

Parameters:



lpFileName

[in] Pointer to a null-terminated string that specifies the name of the object, such as file, COM port, disk device, or console, to create or open.

dwDesiredAccess

[in] Type of access to the object. An application can obtain read-only access, write-only access, read/write access, or device-query access.

dwShareMode

[in] Share mode for the object. If this parameter is set to zero, the object cannot be shared. Subsequent open operations on the object fail until the handle is closed.

This parameter can be set to one or more values.

lpSecurityAttributes

[in] Not used.

dwCreationDisposition

[in] Action to take on files that exist, and which action to take when files do not exist.

dwFlagsAndAttributes

[in] File attributes and flags for the file.

hTemplateFile

[in] Ignored; as a result, this function does not copy the extended attributes to the new file.

Return Value:

An open handle to the specified file indicates success. If the specified file exists before the function call and `dwCreationDisposition` is set to `CREATE_ALWAYS` or `OPEN_ALWAYS`, a call to `GetLastError` returns `ERROR_ALREADY_EXISTS`, even though the function has succeeded. If the file does not exist before the call, `GetLastError` returns zero. `INVALID_HANDLE_VALUE` indicates failure. To get extended error information, call `GetLastError`.



Example:

```
HANDLE hModem;  
hModem = CreateFile( TEXT("COM6:"),  
                    GENERIC_READ | GENERIC_WRITE,  
                    0,  
                    NULL,  
                    OPEN_EXISTING,  
                    0,  
                    NULL);  
if (hModem == INVALID_HANDLE_VALUE)  
    // error opening port; abort
```

Further information on the parameters' values can be found at <http://msdn.microsoft.com/en-us/library/aa914735.aspx>.

3.1.2. WriteFile

This function writes data to a file. WriteFile starts writing data to the file at the position indicated by the file pointer.

Header:

winbase.h

Library:

coredll.lib

Syntax:

```
BOOL WriteFile(  
    HANDLE hFile,  
    LPCVOID lpBuffer,  
    DWORD nNumberOfBytesToWrite,  
    LPDWORD lpNumberOfBytesWritten,  
    LPOVERLAPPED lpOverlapped  
);
```

Parameters:

hFile



[in] Handle to the file to be written to (returned by `CreateFile`). The file handle must have been created with `GENERIC_WRITE` access to the file.

lpBuffer

[in] Pointer to the buffer containing the data to write to the file.

nNumberOfBytesToWrite

[in] Number of bytes to write to the file.

lpNumberOfBytesWritten

[out] Pointer to the number of bytes written by this function call. **WriteFile** sets this value to zero before taking action or checking errors.

lpOverlapped

[in] Unsupported; set to `NULL`.

Return Value:

Nonzero indicates success. Zero indicates failure.

Example:

```
#define BYTES_TO_BE_WRITTEN 4
char atTest[]="AT\r\n";
DWORD written;
HANDLE hModem;
hModem = CreateFile( TEXT("COM6:"),
                    GENERIC_READ | GENERIC_WRITE,
                    0,
                    NULL,
                    OPEN_EXISTING,
                    0,
                    NULL);
if (hModem == INVALID_HANDLE_VALUE)
    // error opening port; abort
if (!WriteFile(hModem, atTest, BYTES_TO_BE_WRITTEN, &written,
NULL))
    // error writing bytes; abort
```

Further information on the parameters' values can be found at <http://msdn.microsoft.com/en-us/library/ms892380.aspx>.

3.1.3. ReadFile

This function reads data from a file, starting at the position indicated by the file pointer.



Header:

winbase.h

Library:

coredll.lib

Syntax:

```
BOOL ReadFile(  
    HANDLE hFile,  
    LPVOID lpBuffer,  
    DWORD nNumberOfBytesToRead,  
    LPDWORD lpNumberOfBytesRead,  
    LPOVERLAPPED lpOverlapped  
);
```

Parameters:

hFile

[in] Handle to the file to be read. The file handle must have been created with `GENERIC_READ` access to the file. This parameter cannot be a socket handle.

lpBuffer

[out] Pointer to the buffer that receives the data read from the file.

nNumberOfBytesToRead

[in] Number of bytes to be read from the file.

lpNumberOfBytesRead

[out] Pointer to the number of bytes read. **ReadFile** sets this value to zero before doing taking action or checking errors.

lpOverlapped

[in] Unsupported; set to NULL.

Return Value:

Nonzero indicates success. Zero indicates failure.

Example:

```
#define BYTES_TO_BE_WRITTEN 4  
#define BYTES_TO_BE_READ 9  
char atTest[]="AT\r\n";  
char atAnswer[10];
```



```

DWORD written;
DWORD read;
HANDLE hModem;
hModem = CreateFile( TEXT("COM6:"),
                    GENERIC_READ | GENERIC_WRITE,
                    0,
                    NULL,
                    OPEN_EXISTING,
                    0,
                    NULL);
if (hModem == INVALID_HANDLE_VALUE)
    // error opening port; abort
if (!WriteFile(hModem, atTest, BYTES_TO_BE_WRITTEN, &written,
NULL))
    // error writing bytes; abort
Sleep(500);
if (!ReadFile(hModem, atAnswer, BYTES_TO_BE_READ, &read, NULL))
    // error reading bytes; abort

```

Further information on the parameters' values can be found at <http://msdn.microsoft.com/en-us/library/ms891445.aspx>.

3.1.4. CloseHandle

This function closes an open object handle.

Header:

winbase.h

Library:

coredll.lib

Syntax:

```

BOOL CloseHandle(
    HANDLE hObject
);

```

Parameters:

hObject

[in] Handle to an open object.

Return value:



Nonzero indicates success. Zero indicates failure.

Example:

```
HANDLE hModem;  
hModem = CreateFile( TEXT("COM6:"),  
                    GENERIC_READ | GENERIC_WRITE,  
                    0,  
                    NULL,  
                    OPEN_EXISTING,  
                    0,  
                    NULL);  
if (hModem == INVALID_HANDLE_VALUE)  
    // error opening port; abort  
CloseHandle(hModem);
```

Further information can be found at
<http://msdn.microsoft.com/en-us/library/ms923946.aspx>.

3.2. PPP Connection

There are several ways to setup a PPP connection for a Windows CE system; in this guide it is described one of the possible methods suitable for headless devices. For further instructions on how to create a subproject refer to MSDN (<http://msdn.microsoft.com/en-us/library/aa913961.aspx>).

- Open Visual Studio 2005 and load your Os Image Design solution.
- In the solution explorer create a Windows CE Console Application subproject with the source files provided by Microsoft in the directory, starting from your Windows CE root (typically WINCE600), \PUBLIC\COMMON\OAK\DRIVERS\NETSAMP\RASENTRY, compile it and add to the OS image. This program adds an entry to the default RAS phonebook from information stored in a configuration file.
- Create the RAS phonebook configuration file (for example called telitHE.ras). In the directory, starting from your Windows CE root (typically WINCE600), \PUBLIC\COMMON\OAK\DRIVERS\NETSAMP\RASENTRY, there is the file *rasentry.txt* that explains the rules for creating this kind of file. Following there is an example:

```
Name=Telit RAS  
UseCountryAndAreaCodes=N  
Phone=[Your provider phone number]  
SpecificIpAddr=N
```



```

SpecificNameServers=N
DeviceType=modem
DeviceName=Telit HSDPA Modem on COM6:
SwCompression=N
IpHeaderCompression=N
DialModifier=[Custom AT commands if needed]
SpecificNameServers=Y
DnsAddr=[Your provider primary DNS address]
AltDnsAddr=[Your provider alternative DNS address]

```

- Create a Windows CE Console Application subproject with the source files provided by Microsoft in the directory, starting from your Windows CE root (typically WINCE600), \PUBLIC\COMMON\OAK\DRIVERS\NETSAMP\RASDIAL compile it and add to the OS image. This program setups the PPP connection according to the selected RAS phonebook entry.
- Start your target and, after the boot, plug the HE. Upload the created RAS phonebook configuration file to the target (for example using the File Viewer tool that you can find in the menu *Target* → *Remote Tools* → *File Viewer*).
- Launch in the target the application *RASENTRY* with the path of the RAS phonebook configuration file as the first argument; you can run an application in your system by choosing the menu voice *Target* → *Run Programs* and selecting the desired application.
- Launch in the target the application *RASDIAL* with the name of the created RAS phonebook entry (Telit RAS in the above example) as the first argument.

If all is successful the PPP connection should be setup: you can launch the *ipconfig* application for checking the target ip address.

3.3. WinCE Limitations

Due to WinCE default configuration, all data or voice connections will be stopped while establishing or disabling a PPP Connection.

In particular when a PPP connection is established while a voice connection is already up, this connection will be interrupted. The same problem occurs when disconnecting a PPP connection during a voice connection.

In order to avoid this effect, it is suggested to add the following registry key in the project registry file:

```

[HKEY_LOCAL_MACHINE\Drivers\Unimodem\Settings]
"Reset"="AT<cr>"

```

