USB PC Drivers Based on Generic HID Class

- **Supported by Windows 98® SE or later**
- **Full Duplex Communication**
- **Send Commands Through the EP 0**
- **Dynamic Link Library Supported by any Compiler: VC++, JAVA, VB...**
- **Auto-detection of device for VC++ application**
- **Point-to-Point Communication**

1. Introduction

This application note describes how to integrate the USB HID DLL in your application. The provided examples are based on VC++ and JAVA compilers, however the DLL can be used with any compiler (VB, Delphi, LabView...).

Simple code examples that demonstrate different types of implementation are given.

USB Microcontrollers

Application Note

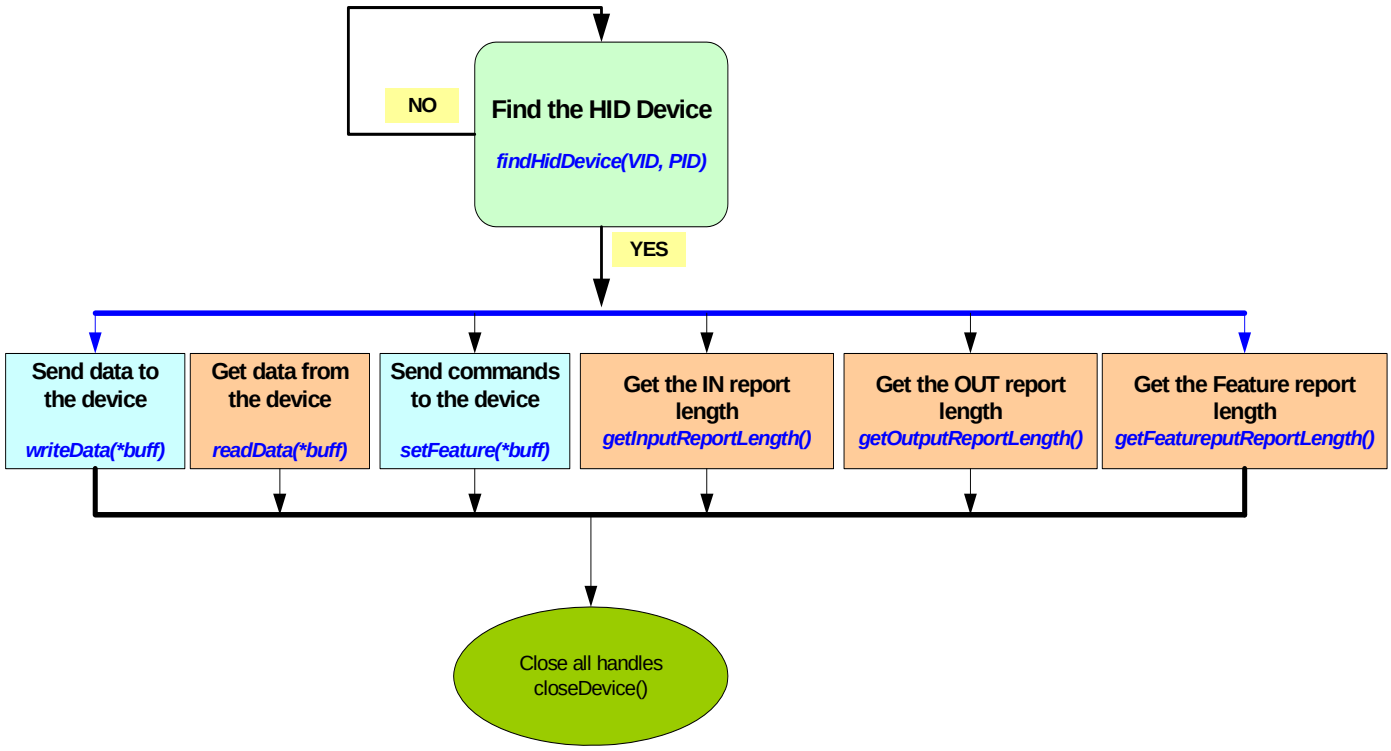
7645B–USB–07/08

2. DLL functions

As specified in the USB HID specification, the Generic HID application uses two reports (IN/OUT) to send and receive data. The length of these reports is assigned in the firmware and automatically detected by the DLL following the firmware setting (please refer to the *USB Generic HID Implementation* application note to see how to modify these values if needed).

Please note that this DLL allows you to communicate with one Generic HID device and only one device at a time. You cannot manage several devices at the same time using this DLL.

Figure 2-1. DLL functions



.

2.1 findHidDevice

This function (BOOLEAN) allows to find the Generic HID device using the vendor ID (VID)/the product ID (PID) and open a handle if the device is connected.

Input

const UINT VendorID: this is the vendor ID

const UINT ProductID: this is the product ID

Output

FALSE: if the device is not found. More information can be found using GetLastError().

GetLastError will return:

ERROR_USB_DEVICE_NOT_FOUND if Device is cannot be found.

ERROR_USB_DEVICE_NO_CAPABILITIES if device found but capabilities cannot be retrieved.

TRUE: if the connection has succeeded and the handle is opened.

2.2 closeDevice

Closes the communication with the USB device and all related handles.

2.3 writeData

This function (BOOLEAN) sends data to the device (OUT data). The maximum data length supported by this function must be lower or equal to the value given by the function getOutputReportLength (see Section 2.9 on page 5).

If the data length exceeds the maximum length specified in the firmware, the user has to send data in several packets.

When data length is lower than the maximum length, this function will complete the remaining bytes with zero (0x00).

Input

UCHAR* buffer: pointer of the packet to be sent.

2.3.1 Output

FALSE: if data transmission fails. GetLastError() will return ERROR_WRITE_FAULT code

TRUE: if the packet was successfully transferred.

2.4 readData

This function (BOOLEAN) read the data packets sent by the device (IN data). To avoid data loss this function should be called in continuous mode (using a thread or a timer).

.

Input

UCHAR* buffer: Pointer to the buffer which will contain the received packet.

The buffer must have the length of the IN report given by the getInputReportLength function (see Section 2.10 on page 5).

2.4.1 Output

FALSE: if no data is available.

TRUE: if data are received and stored in the buffer.

2.5 setFeature

This function (BOOLEAN) allows the user to send a command data to control the HID device (i.e.: Start the bootloader, start a new task...). Data will be transmitted over the endpoint 0 as a "*SetReport*" request (Refer to the HID Specification for further information). The endpoints IN and OUT will be used for the applicative raw data transfer only.

The data length is fixed by the firmware and can be obtained using the function getFeatureReportLength (please refer to the Section 2.11 on page 5). The data length must not exceed the length returned by getFeatureReportLength function.

Input

UCHAR* buffer: Pointer to the buffer which contains the received packet.

Output

FALSE:if data transmission fails.

TRUE: if data are well transferred.

2.6 hidRegisterDeviceNotification

Please note that this function can be used only with VC++ project.

This function notifies the application if a new plug & play device has been plugged or unplugged.

Input

HWND hWnd - Handle to a window.

Output

FALSE: if the function fails. To get extended error information, call GetLastError.

TRUE: if the function succeeds.

2.7 hidUnregisterDeviceNotification

Please note that this function can be used only with VC++ project.

This function closes the specified device notification handle.

Input

HWND hWnd - Handle to a window.

Output

FALSE: if the function fails. To get extended error information, call GetLastError.

TRUE: if the function succeeds.

2.8 isMyDeviceNotification

Please note that this function can be used only with VC++ project.

This function allows to check if the new device (plugged or unplugged) notified by ***"hidRegisterDeviceNotification"*** is the used HID device or not.

Input

DWORD dwData, the value given by OnDeviceChange 2nd parameters

Output

TRUE: if the device connected/disconnected is the used HID device

FALSE: if this is another device

2.9 getOutputReportLength

This function allows the user to get the length of the OUT report (data packet sent from the PC to the device). This value is specified in the firmware.

2.10 getInputReportLength

This function allows the user to get the length of the IN report (data packet sent from the device to the PC). This value is specified in the firmware.

2.11 getFeatureReportLength

This function allows the user to get the length of the Feature report (Control data packet sent from the PC to the device). This value is specified in the firmware.

3. PC demos

3.1 VC++ demo

The VC++ demo allows the user to see how to load the AtUsbHid.dll in a project, and also how to use the plug & play notification.

3.1.1 Load the DLL in Visual C++ Application

The file AtUsbHid.h provides the macros which help to load and use the functions present in the Atmel USB HID DLL.

When designing an application using the DLL you need to do the following:

- create a handler for the DLL: **HINSTANCE hLib = NULL;**
- Load the DLL using the function **hLib =LoadLibrary(AT_USB_HID_DLL);**
- Load each DLL functions using **loadFuncPointers(hLib)**

Once these steps have been performed without error, the DLL and its functions are loaded in your application and can be called using the macro **DYNCALL(**DllFunction()**).**

When the application is stopped, it is convenient to free the DLL from memory using the function **FreeLibrary(hLib).**

You must ensure that USB device handle has been closed before freeing the DLL from memory.

3.1.2 Using Automatic Device Connection/Disconnection Feature

The DLL provides the functions which allow the user to detect the connection/disconnection of the device.

To perform this feature you have to do the following actions:

Register you application to get device change notification using: **DYNCALL(hidRegisterDeviceNotification)((m_hWnd)).**

Add the function **ON_WM_DEVICECHANGE()** in your Message Map application.

Creates a function called **OnDeviceChange(UINT nEventType, DWORD dwData)** which will be called each time a device status changes.

In the function **OnDeviceChange**, call the function **DYNCALL(isMyDeviceNotification(dwData))** to know if the status of your device has changed (connected or disconnected). (See code demo code in UsbHidDemoCodeDlg.cpp)

When exiting the application, it is convenient to unregister it using the function: **DYNCALL(hidUnregisterDeviceNotification(m_hWnd)).**

3.1.3 Using readData

As data can be sent continuously by the device. It's interesting to read data using a timer base function. This allows you to poll continuously the readData function.

To do so, you have to do the following:

Add the function **ON_WM_TIMER()** in your Message Map application.

Create a function **OnTimer(UINT nIDEvent)** which will call the function **DYNCALL(readData(sbuffer)**.

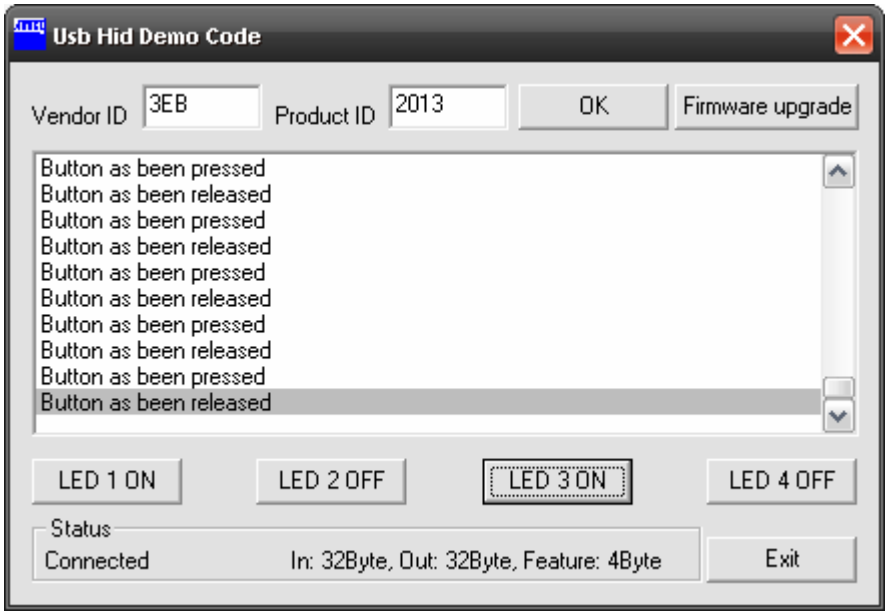
Now you can set the Timer for a specified interval using **SetTimer(n,x,y)**; to call the readData function each *x* ms when your device is connected.

Kill the timer using **KillTimer(n)** when your device is disconnected.

3.1.4 User Interface

Hereunder is a screen shot of the provided demo. Please note that the default PID is related to one specific demo (Atmel demos which have a Generic HID interface do not have the same PID). You may have to modify this PID parameter to match with the device you are using (refer to the firmware or the device manager to get the VID/PID used by your demo)

Figure 3-1. VC++ based demo



Hereunder is the description of the GUI components:

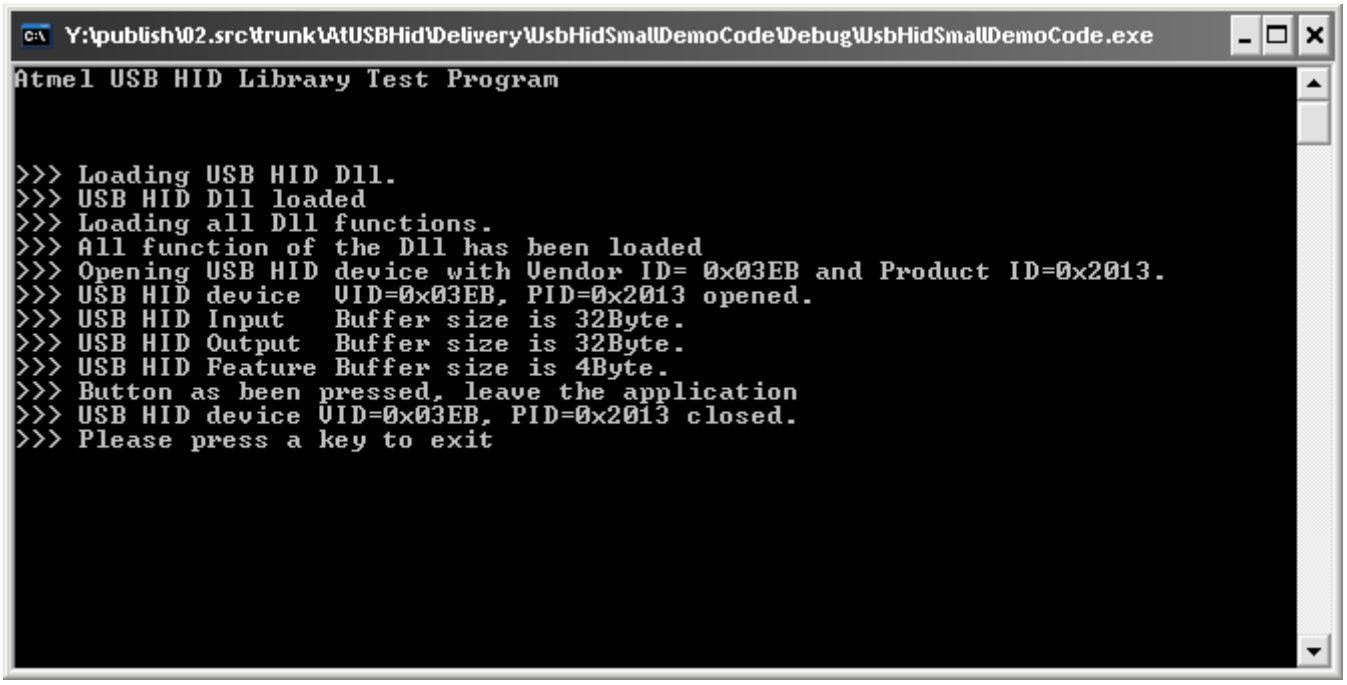
- The *Vendor ID*, *Product ID* box are used to specify the VID/PID of the device.
- *OK* button should be pushed once the VID/PID are correctly set.
- LED 1...LED4 button are used to switch ON/OFF the LEDs of the board.
- Firmware Upgrade button allows the user to start the bootloader to upgrade the firmware through the USB interface (Refer to the bootloader datasheet for further details).

- Exit button closes the application
- Status field gives the connection state and also when the device is connected gives the lengths of the IN report, OUT report and Feature report (these parameters will be automatically used by the DLL to send/receive data)

3.1.5 DOS demo

This demo gives a simple console application example. This demo uses a fixed VID/PID and has to be recompiled to modify these parameters. The device have to be connected and running with the Generic HID firmware before performing this console application.

Figure 3-2. DOS Interface



Note: This project can be compiled using the MinGw (www.**mingw**.org). The command line is:

```
mingw32-g++ -O2 -Wall  UsbHidSmallDemoCode.cpp -o AtUsbHidMinGw.exe -I.
```

3.2 JAVA demo

The JAVA demo allows the user to see how to integrate the AtUsbHid.dll in a JAVA project.

The interface between the *AtUsbHid.dll* and the JAVA is done through the package *AtUsbHidJni.jar.*

3.2.1 AtUsbHid.dll integration

To integrate the *AtUsbHid.dll you* have to follow the steps below:

- Add the following code in the import section of your JAVA file:

```
import com.atmel.atusbhidjni.AtUsbHidJni
```

- Create a new object to use the DLL:

```
AtUsbHidJni usbDevice = new AtUsbHidJni();
```

- Load the DLL:

```
usbDevice.loadLibraryUsbHid();
```

- Now, the DLL is ready for use. Please refer to the DLL functions section for further details regarding the DLL functionalities.
- Before existing the application, it is important to unload the DLL:

```
usbDevice.UnloadloadLibraryUsbHid();
```

- To compile the project, please add to the class path of the *AtUsbHidJni.jar* package:

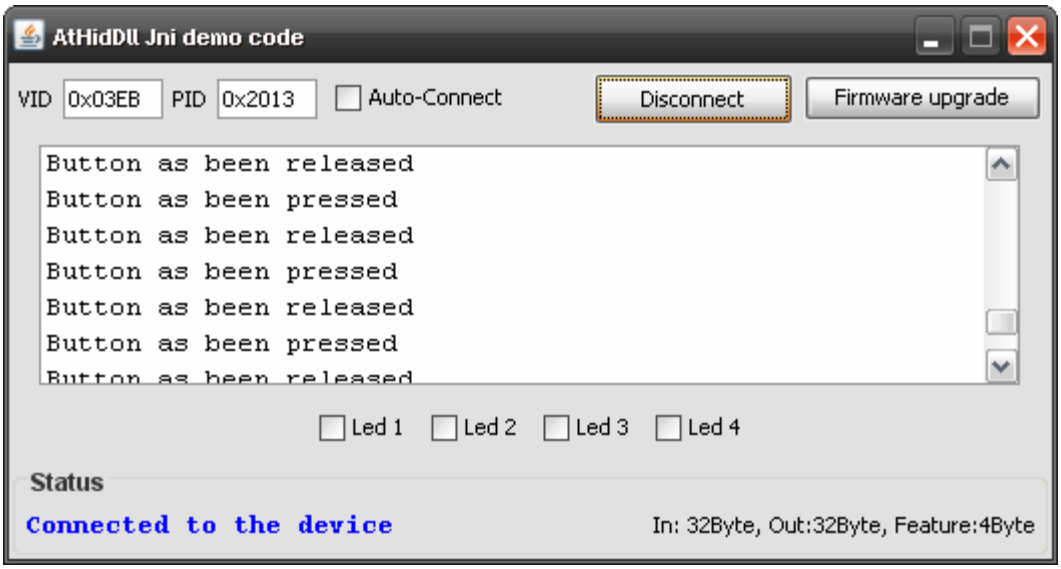
```
JAVAc userhid.JAVA -classpath AtUsbHidJni.jar
```

Note: Please refer to the HTML documentation provided with the DLL package for further information.

3.2.2 User interface

The GUI source code is available in the JNICodeForHIDDLL folder. Hereunder is the JAVA user interface:

Figure 3-3. JAVA User interface

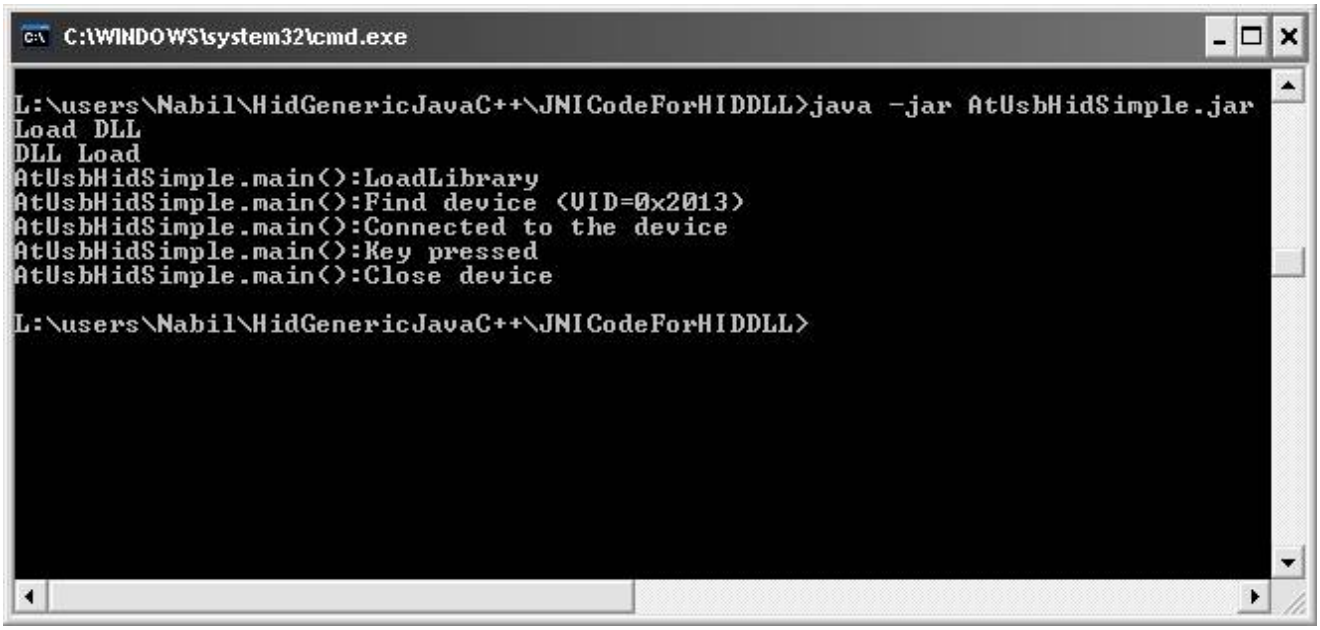


The components have the same roles as described for the VC++ interface (refer to section 3.1.4). The Auto-Connect box is used to allow the application to detect automatically the connection/disconnection of the device.

3.2.3 DOS demo

This demo gives a simple console application example. The demo uses a fixed VID/PID and has to be recompiled to modify these parameters. The device has to be connected and running with the Generic HID firmware before performing this console application.

Figure 3-4. DOS Interface



4. The package architecture

When you unzip the DLL package, you'll find several folders. Hereunder is the content of each one:

4.1 AtUsbHid

This folder contains the AtUsbHid.dll and the AtUsbHid.h files.

4.2 ExeDemo

This folder contains the different executable demo examples.

4.3 JNICodeForHIDDLL

This folder contains the source code of the JAVA project.

4.4 UsbHidDemoCode

This folder contains the source code of the VC++ project.

4.5 UsbHidSmallDemoCode

This folder contains the source code of the VC++ small demo (DOS demo).

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
8051@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2008 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.