AVR270: USB Mouse Demonstration

Features

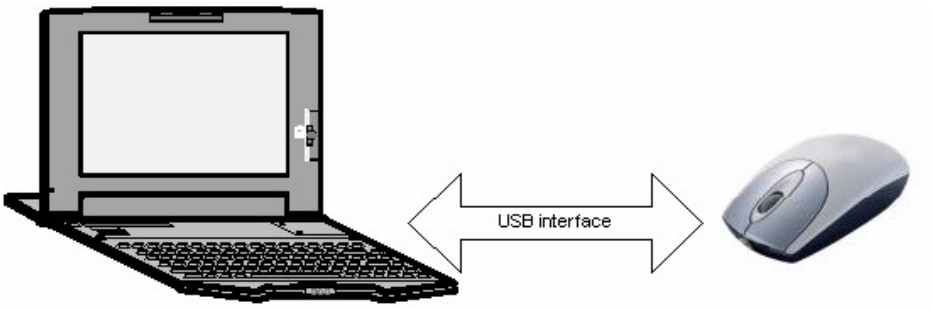
- **Runs with AT90USB Microcontrollers at 8MHz**
- **USB Low Power Bus Powered Device (less then 100mA)**
- **Supported by any PC running Windows® (98SE or later), Linux® or Mac OS®.**
- **Less than 3Kbytes of Code Required**
- **X, Y Movement, Left Button Supported**

1. Introduction

The PS/2 interface has disappeared from the new generations of PCs to leave the place to the USB interface. This change has to be followed by the designers of pointing devices, who should integrate the USB interface and allow an easy connection to new PCs.

This document describes a simple mouse project. It allows to quickly test USB hardware using AT90USB without any driver installation.

A familiarity with *USB Software Library for AT90USBxxx Microcontrollers* (doc 7675, included in the CD-ROM & Atmel website) and the HID specification (http://www.usb.org/developers/hidpage) is assumed.



8-bit AVR® Microcontrollers

Application Note

7604B–AVR–03/08

2. Hardware Requirements

The USB mouse application requires the following hardware:

- AVR USB evaluation board (STK525, AT90USBKey, STK526...or your own board)
- AVR USB microcontroller
- USB cable (Standard A to Mini B)
- PC running on Windows (98SE, ME, 2000, XP), Linux or MAC OS with USB 1.1 or 2.0 host

3. In-System programming and Device Firmware Upgrade

To program the device you can use the following methods:

- The JTAG interface using the JTAGICE MKII
- The SPI interface using the AVRISP MKII
- The USB interface thanks to the factory DFU bootloader and Flip software
- The parallel programming using the STK500 or STK600

Please refer to the hardware user guide of the board you are using (if you are using Atmel starter kit) to see how to program the device using these different methods.

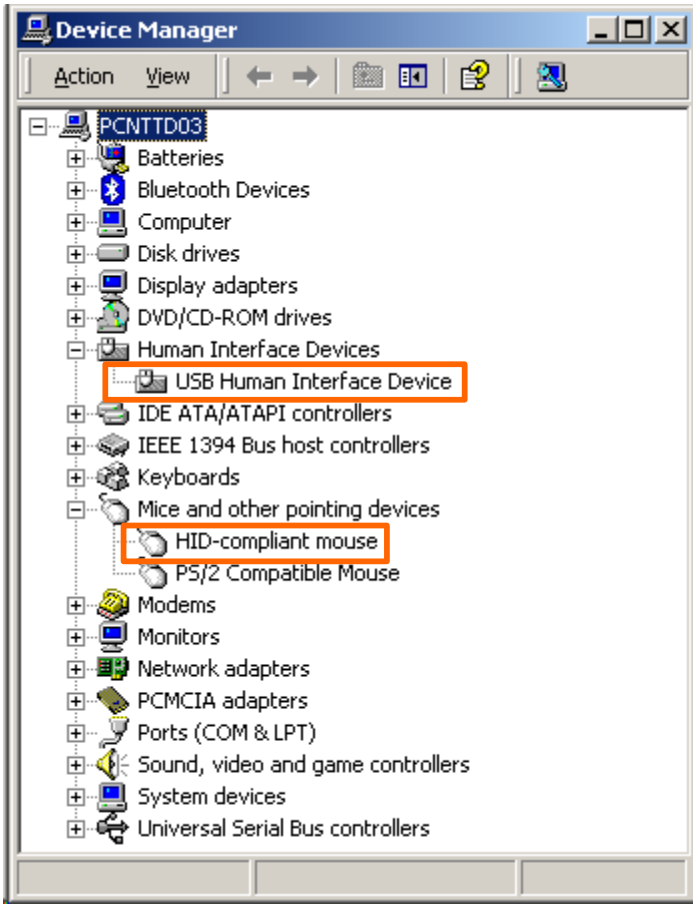
Please refer to Flip[1] help content to see how to install the USB driver and program the device through the USB interface.

Note: 1. Flip is software provided by atmel to allow the user to program the AVR USB devices through the USB interface (No external hardware required) thanks to the factory DFU bootloader.

4. Quick Start

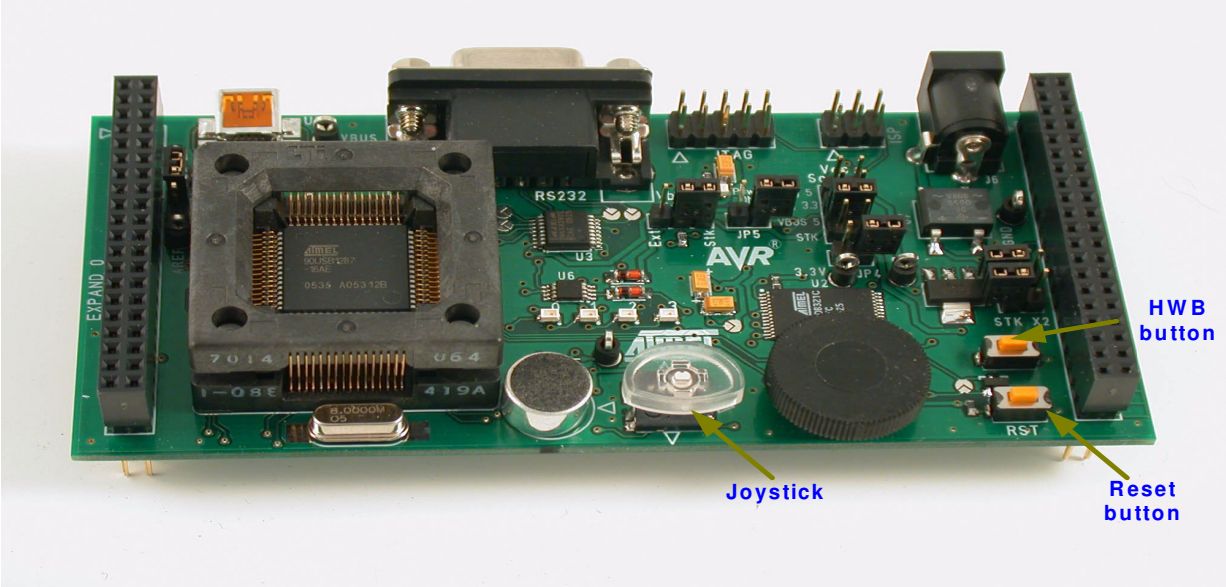
Once your device is programmed with *usb_mouse.a90* file, you can start the mouse demonstration. Check that your device is enumerated as a mouse (see figure 11), then you can use the kit as a mouse.

Figure 4-1. Mouse Enumeration



The figure below shows the STK525 used by the demo (you may use another kit: AT90USBKey, STK526, depending on the AVR USB product you are working with):

Figure 4-2. Demonstration Board



To move the mouse pointer in several directions (up, down, left, right), you have to move the joystick. The HWB button will be used as a left button.

5. Application Overview

The USB mouse application is a simple data exchange between the PC and the mouse.

The PC asks the mouse if there is new data available each *P* time (polling interval time), the mouse will send the data if it is available, otherwise it will send a NAK (No Acknowledge) to tell the PC that there is no data available.

Data sent to the PC are called ‘report’. This report has the structure below:

Figure 5-1. USB Report Structure

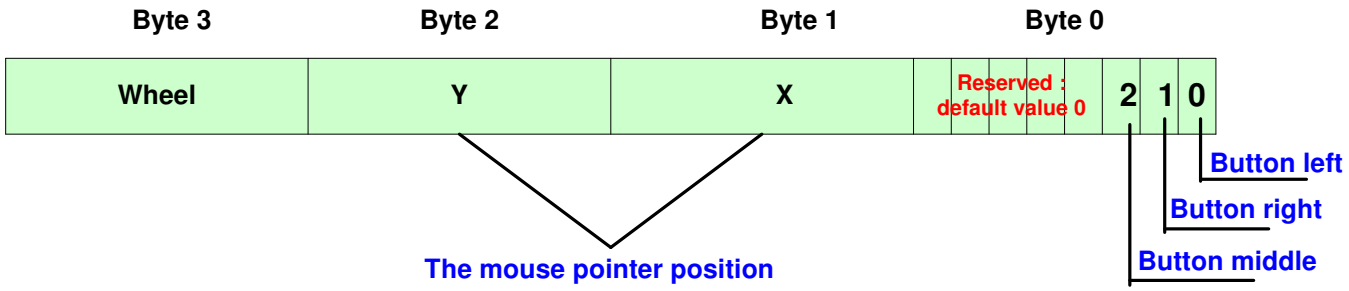
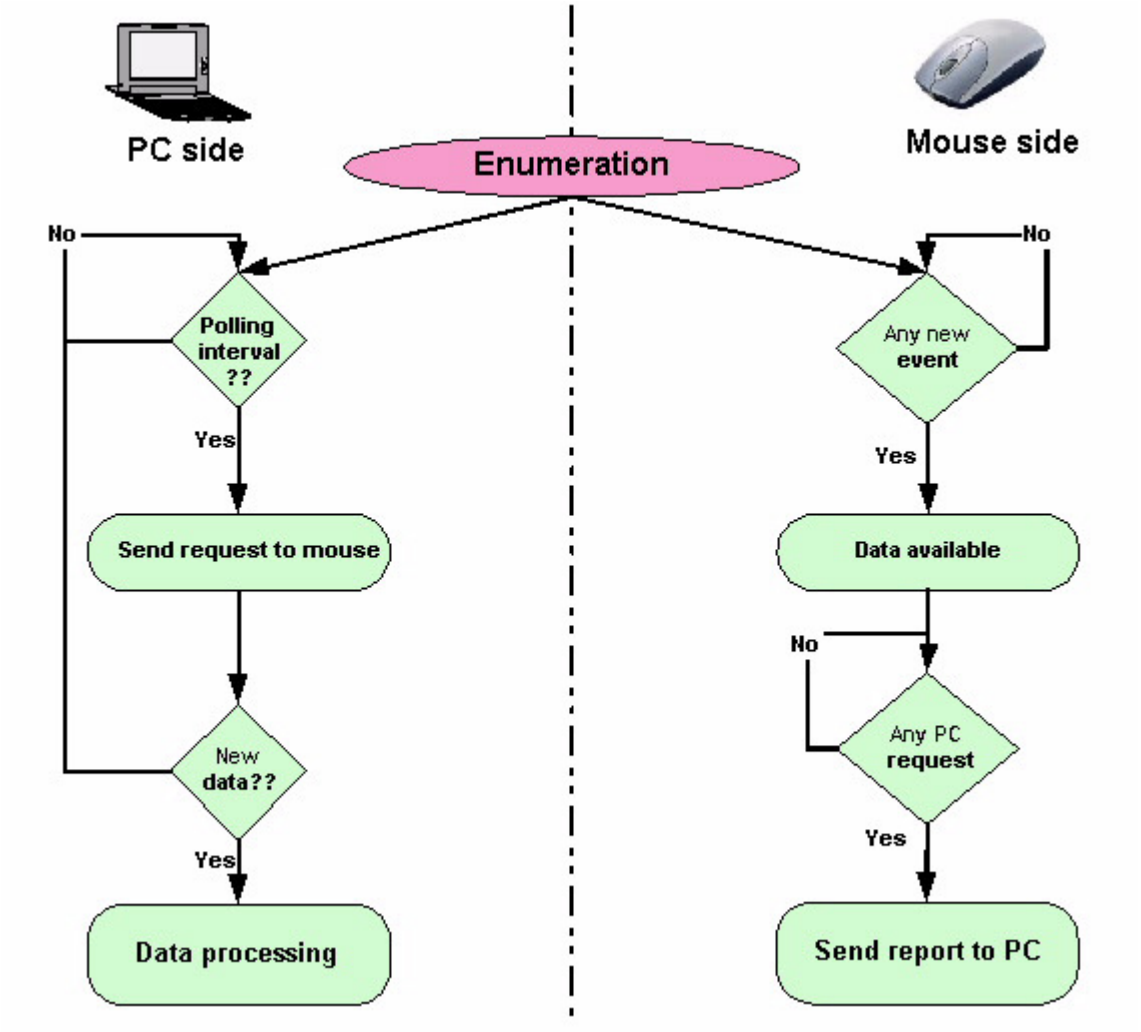


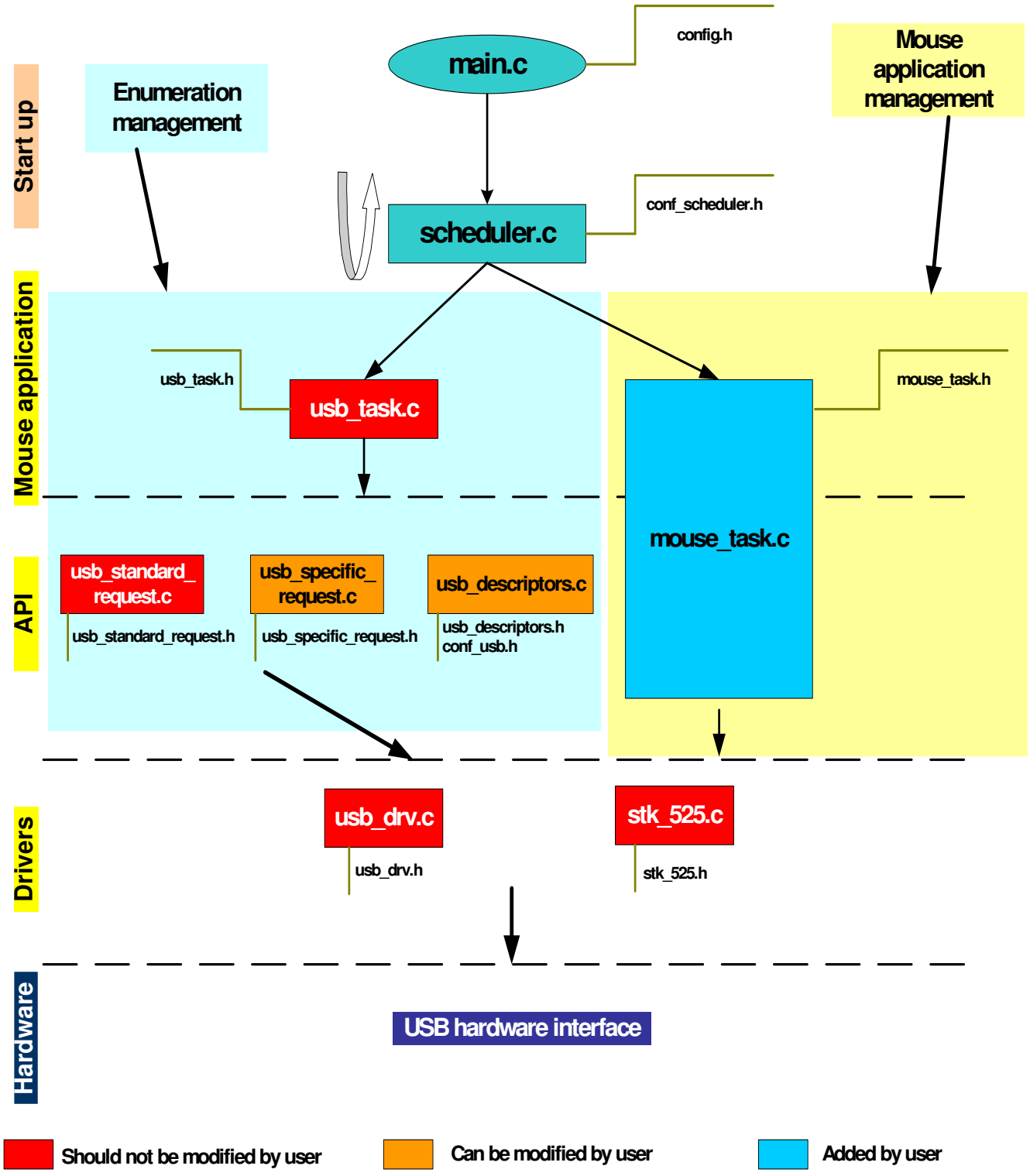
Figure 5-2. Application Overview



6. Firmware

As explained in the *USB Software Library for AT90USBxxx Microcontrollers* document (doc 7675) all USB firmware packages are based on the same architecture.

Figure 6-1. USB Mouse Firmware Architecture

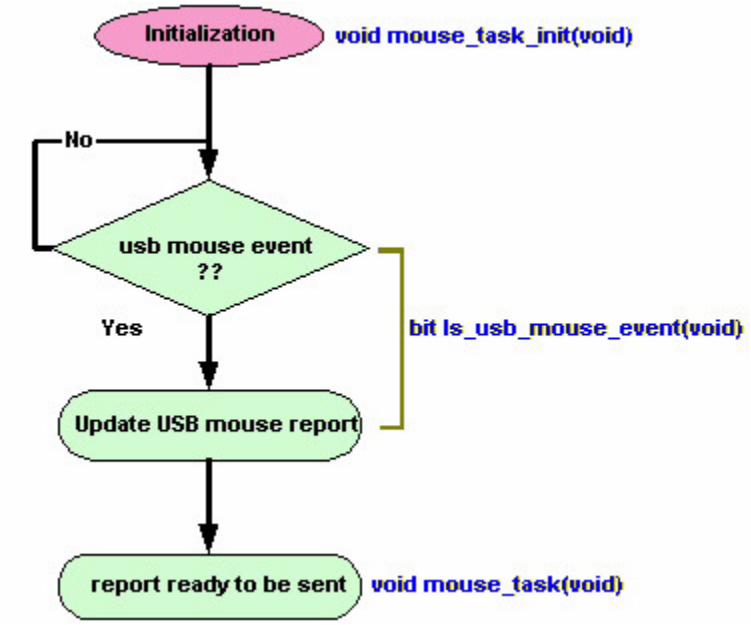


This section is dedicated to the mouse module only. The customization of the files described hereafter allow the user to build his own mouse application.

6.1 mouse_task.c

This file contains the functions to initialize the mouse, collect the report data and put it in the endpoint FIFO to be ready to be sent to the PC.

Figure 6-2. Mouse Application



6.1.1 mouse_task_init

This function performs the initialization of the mouse parameters and hardware resources (joystick, button...).

6.1.2 ls_usb_mouse_event

If a mouse event occurs, this function updates the USB mouse report and returns true. Otherwise it returns false.

6.1.3 mouse_task

This function checks if *ls_usb_mouse_event* is true and loads the report in the USB endpoint FIFO to be transmited to the host.

6.2 stk_52x.c.

This file contains all the routines to manage the STK 52x board resources (Joystick, potentiometer, Temperature sensor, LEDs...). The user should not modify this file when using the STK52x board. Otherwise he has to build his own hadware management file.

6.3 How to manage the Remote Wake Up feature

The *Remote Wake Up* is an optional feature specified by the USB to allow the device to wake the host up from a stand by mode (refer to the USB specification for further details). This request is the only request which can be initiated by the device, but it has to be allowed by the host. The host sends a Set Feature request to enable the Remote Wake Up feature just before sending the suspend request. If the host did not send the Set Feature (RemoteWakeUpEnable), the device is not allowed to perform this feature.

A USB device reports its ability to support remote wakeup in its configuration descriptor (refer below to see how it is done with Atmel library). If a device supports remote wakeup, it must also be allowed the capability to be enabled and disabled using the standard USB requests.

The configuration descriptor is defined in the usb_descriptors.h file as below:

```
// HID Mouse CONFIGURATION
#define NB_INTERFACE       1
#define CONF_NB            1
#define CONF_INDEX         0
#define CONF_ATTRIBUTES    USB_CONFIG_BUSPOWERED
#define MAX_POWER          50          // 100 mA
```

To setup the Remote Wake Up feature, you have to modify the `CONF_ATTRIBUTES as below:`

```
#define CONF_ATTRIBUTES    (USB_CONFIG_BUSPOWERED|USB_CONFIG_REMOTEWAKEUP)
```

If the device supports the Remote Wake Up feature, the user has to manage the Set_Feature(DEVICE_REMOTE_WAKEUP) request using the *void usb_set_feature(void).*

Once the Set_Feature(DEVICE_REMOTE_WAKEUP) is well managed, you can use any button (must be used in external interrupt/pin change mode) for example to wake up the host. To do this, you have to take care of the following details:

- First, the USB controller must have detected the “suspend” state of the line: the remote
- wake-up can only be sent when a SUSPI flag is set.
- The firmware has then the ability to set RMWKUP to send the “upstream resume” stream.
- This will automatically be done by the controller after 5ms of inactivity on the USB line.
- When the controller starts to send the “upstream resume”, the UPRSMI interrupt is triggered
- (if enabled). SUSPI is cleared by hardware.
- RMWKUP is cleared by hardware at the end of the “upstream resume”.
- If the controller detects a good “End Of Resume” signal from the host, an EORSMI interrupt

is triggered (if enabled).

6.4 How to modify my device from non-bootable to bootable device

Please note that HID device may be bootable or non-bootable. By default, the HID demo provided by Atmel are non-bootable devices. If your application needs to be bootable, you have to modify the *sub-class* parameter (*usb_descriptors.h*):

```
// USB Interface descriptor Keyboard
#define INTERFACE_NB_MOUSE 0
#define ALTERNATE_MOUSE 0
#define NB_ENDPOINT_MOUSE 1
#define INTERFACE_CLASS_MOUSE 0x03    // HID Class
#define INTERFACE_SUB_CLASS_MOUSE 0x00 // Non-bootable
#define INTERFACE_PROTOCOL_MOUSE 0x01 //Keyboard
#define INTERFACE_INDEX_MOUSE 0
```

Set the ***INTERFACE_SUB_CLASS_MOUSE*** to 1 to convert the mouse to a bootable device.

7. PC Software

The USB mouse application does not require any PC software.

8. Limitations

The middle and the right buttons are not supported by this demonstration.

9. Related Documents

AVR USB Datasheet (doc 7593)

USB Software Library for AT90USBxxx Microcontrollers (doc 7675)

USB HID class specification (www.usb.org)

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
Enter Product Line E-mail

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.