Atmel AVR4904: ASF - USB Device HID Keyboard Application

Features

- USB 2.0 compliance
  - Chapter 9 compliance
  - USB HID compliance
  - Low Speed (1.5Mbit/s), Full Speed (12Mbit/s) data rates
- Standard USB HID keyboard implementation
  - Key
  - Modifier keys (shift, CTRL, …)
  - LED report
- Remote wakeup support
- USB bus powered support
- Real time (O.S. compliance, interrupt driven)
- Support 8-bit and 32-bit AVR®

1 Introduction

The aim of this document is to provide an easy way to integrate a USB keyboard device application on a new or existing project.

Atmel Microcontrollers

Application Note

Rev. 8446A-AVR-10/11

2 Abbreviations

ASF:	AVR Software Framework
CD:	Composite Device: a USB device with more than one interface
FS:	USB Full Speed
HID:	Human Interface Device
HS:	USB High Speed
LS:	USB Low Speed
UDC:	USB device Controller
UDD:	USB device Descriptor
UDI:	USB device Interface
USB:	Universal Serial Bus
SOF:	Start of Frame
ZLP:	Zero Length Packet

3 Overview

This document includes four sections for all types of requirements when building a USB device HID keyboard application:

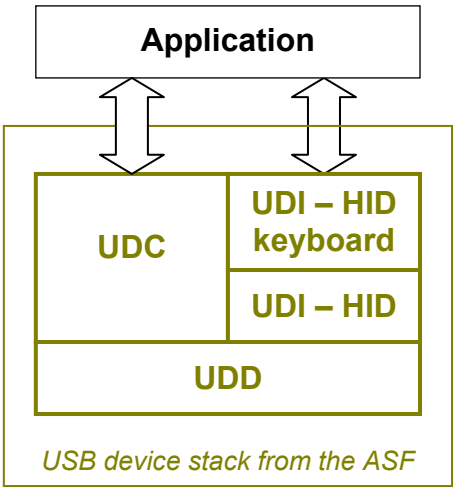
- **Quick start**
  Describes how to start a ready to use HID keyboard device example
- **Example description**
  Describes a HID keyboard device example
- **Building a USB device keyboard**
  Describes how to add a USB device keyboard interface in a project
- **Keyboard in a USB Composite Device**
  Describes how to integrate a keyboard interface in a composite device project

For all these sections, it is recommended to know the main modules organization of a HID keyboard application:

- User Application
- USB device Interface HID keyboard (UDI-HID keyboard)
- USB device Interface HID (UDI-HID)
- USB device Controller (UDC)
- USB device Driver (UDD)

For more advanced information concerning the USB stack implementation, please refer to the application note Atmel® AVR4900: ASF - USB Device stack.

Figure 3-1. USB device HID keyboard architecture.



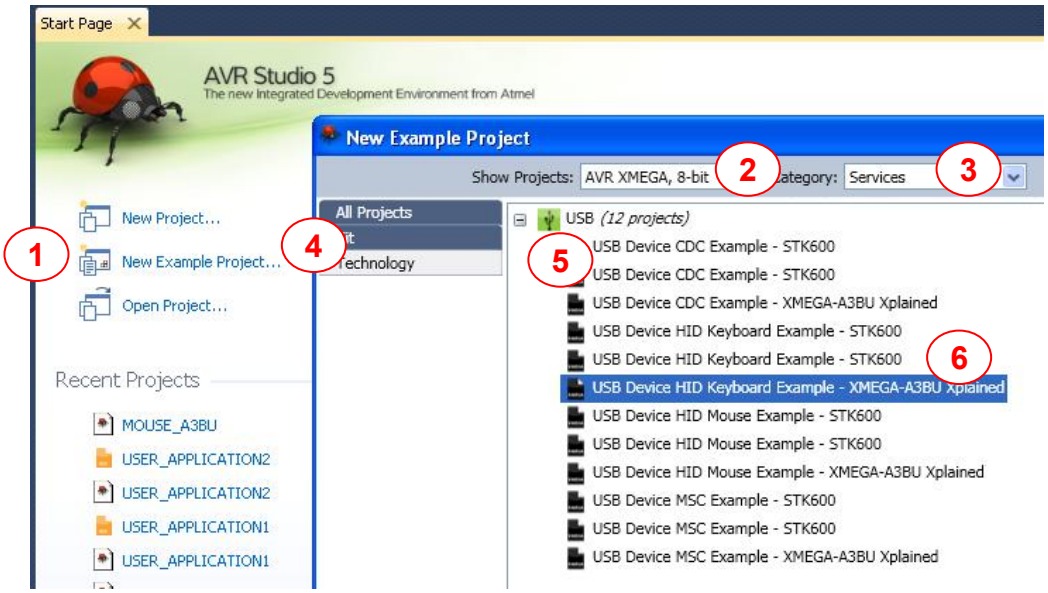
NOTE

The USB device stack is available in the Atmel AVR Software Framework (ASF) in the common/services/usb directory.

4 Quick start

The USB device keyboard examples are available in Atmel AVR Studio® 5 and ASF. It provides a keyboard peripheral application.

1. Board powering and USB connection.
   Connect a USB cable between board and the USB Host, this cable powers the board.
2. AVR Studio 5 allows the creation of a New Example Project.
   In the examples list, select a USB device HID Keyboard Example corresponding to the Atmel board used. Use the filter list to find quickly the example.

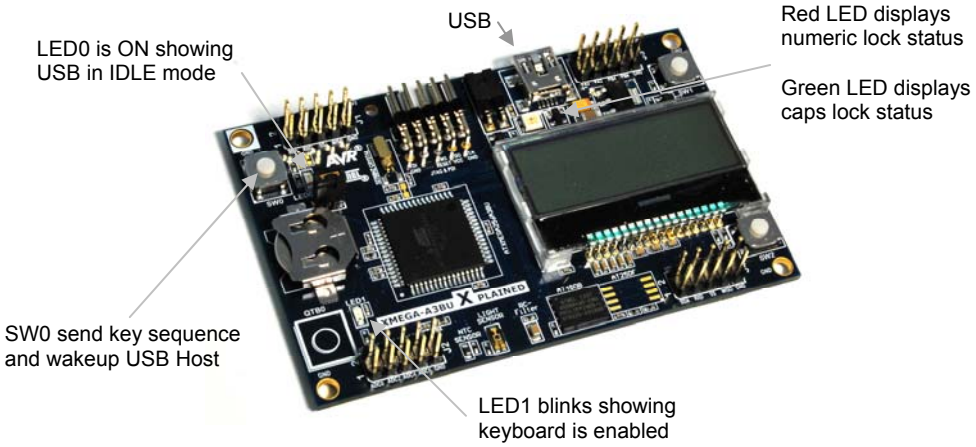


3. Compile, load and execute.
   The project does not require any modification and only needs to be compiled, loaded and run. Connect the Atmel debugger supported by the board and press F5. Thus, a USB keyboard is available on USB Host.
4. Use it.
   When a button is pressed, the example sends a key sequence which opens a note pad application under Windows® O.S. and a key sequence “Atmel AVR USB Keyboard”.

Also the user interface of the board can provide the following information:

- A LED is ON when USB device is in IDLE mode and is OFF in SUSPEND mode.
- A LED blinks showing that keyboard interface is enabled by the USB Host.
- A LED is ON when the numeric keys is locked.
- A LED is ON when the capital key is locked.

Figure 4-1. Example with Atmel XMEGA® A3BU Xplained board.



The user interface description (specific to the board) is defined at the end of the `ui.c` source file. This file is available inside the project folder under “common/services/usb/class/hid/device/kbd/example/part_board/”.

5 Example description

The Atmel AVR Software Framework (ASF) provides a USB device keyboard example for various Atmel AVR products.

All these examples share common files.

Files description and execution behavior are described later in this section.

5.1 Example content

Table 5-1 introduces a summary of the main files included in the USB device HID keyboard example. These files are associated to the modules described in **Figure 3-1.** USB device HID keyboard architecture.

Table 5-1. USB device HID keyboard example files.

Modules	Files	ASF paths	Description

Application	`main.c`
	`ui.c`
	`conf_usb.h`
		Examples folder	Main loop
			Set up hardware switches to keyboard operations
			USB device configuration

UDI HID keyboard	`udi_hid_kbd.c/h`	common/services/usb/class/hid/device/kbd/	Standard HID keyboard Class implementation
	`udi_hid_kbd_desc.c` `udi_hid_kbd_conf.h`	common/services/usb/class/hid/device/kbd/	USB descriptors for an USB device with HID keyboard interface (not applicable for USB composite device)
UDI HID	`udi_hid.c/h`	common/services/usb/class/hid/device/	Common HID class implementation
	`usb_protocol_hid.h`	common/services/usb/class/hid/	HID protocol constants

UDC	`udc.c/h`
	`udc_desc.h`
	`udi.h`
	`udd.h`
		common/services/usb/udc/	USB device core

	`usb_protocol.h` `usb_atmel.h`	common/services/usb/	USB protocol constants

UDD	`usbb_device.c/h`
	`usbc_device.c/h`
	`usb_device.c/h`
		avr32/drivers/usbb/
		avr32/drivers/usbc/
		xmega/drivers/usb/
			USB drivers

5.2 Example behavior

The `main.c` and `ui.c` files implement the user interface of the HID keyboard application.

It is comprised of four steps:

1. Start USB device.
```
udc_start();
udc_attach(); // Must be called when the USB cable is plugged
              // Cable plugged is detected via VBus events
```
2. Wait the enable of HID keyboard interface via callback.
```
UDI_HID_KBD_ENABLE_EXT() // Authorize the HID keyboard events
```
3. Scan the switches and send a key sequence in SOF interrupt.
```
if(is_button_pressed()) {
   // start key sequence
```

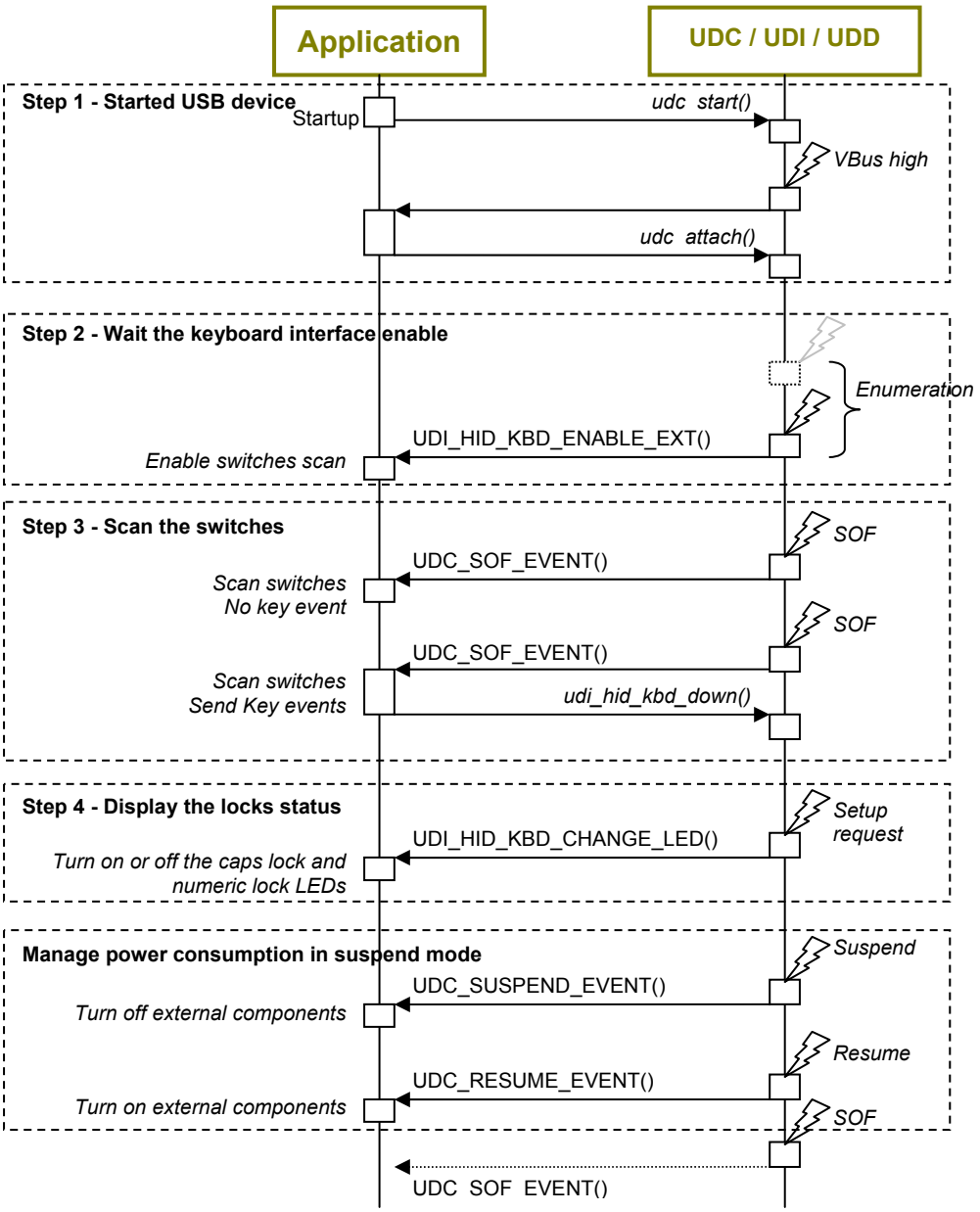
```
    Sequence_running = true;
}
// Sequence process running each 200ms
if (200 > cpt_sof) {
    return;
}
cpt_sof = 0;

if (sequence_running) {
    // Extract the next key of the array sequence
    // Send this key with a following function
    udi_hid_kbd_modifier_down(modifier_key_value); // e.g. "shift"
    udi_hid_kbd_modifier_up(modifier_key_value); // e.g. "shift"
    udi_hid_kbd_down(key_value);
    udi_hid_kbd_up(key_value);
}
```

Tip: To simplify the implementation, the SOF event is used to scan switches each 1ms.

4. Display the new caps and numeric lock status via callback.
   `UDI_HID_KBD_CHANGE_LED(HID_LED_NUM_LOCK | HID_LED_CAPS_LOCK)`

Figure 5-1. Example behavior sequence.



6 Building a USB device keyboard

The USB device keyboard modules are available in Atmel AVR Studio 5 and can be imported in an AVR Studio 5 project.

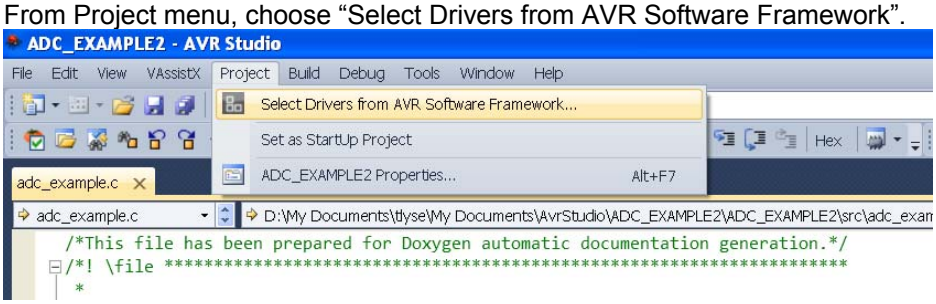
This section describes how to add a USB device keyboard in a project:

1. Import USB keyboard module.
2. Configure personal USB parameters.
3. Call USB routines to run USB device.

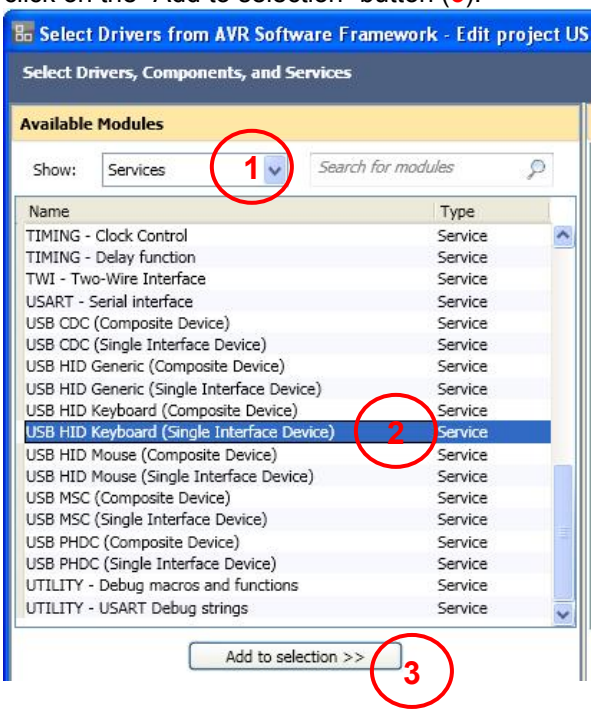
6.1 Import USB module

To import the USB HID keyboard module, follow the instructions below:

1. Open or create your project:
2. From Project menu, choose “Select Drivers from AVR Software Framework”.



3. Select Services (1), choose USB HID Keyboard (Single Interface Device) (2) and click on the “Add to selection” button (3).



6.2 USB configuration

All USB stack configurations are stored in the `conf_usb.h` file in the application module. These configurations are simple and do not require any specific USB knowledge.

There is one configuration section for each USB modules: UDC, UDI and UDD.

The UDC configuration possibilities are described in the application note AVR4900: ASF – USB device stack, Section 7.1.1 - USB device configuration.

The UDD configuration possibilities are described in the application note AVR4900: ASF – USB device stack, Section 7.1.3 - USB Drivers configuration.

The UDI which is the keyboard interface does not require any configuration.

IMPORTANT

It is important to verify the configuration defined in the `conf_clock.h` file, because the USB hardware requires a specific clock frequency (see comment in the `conf_clock.h` file).

6.3 USB implementation

This section describes source code to add and to run a USB device keyboard application.

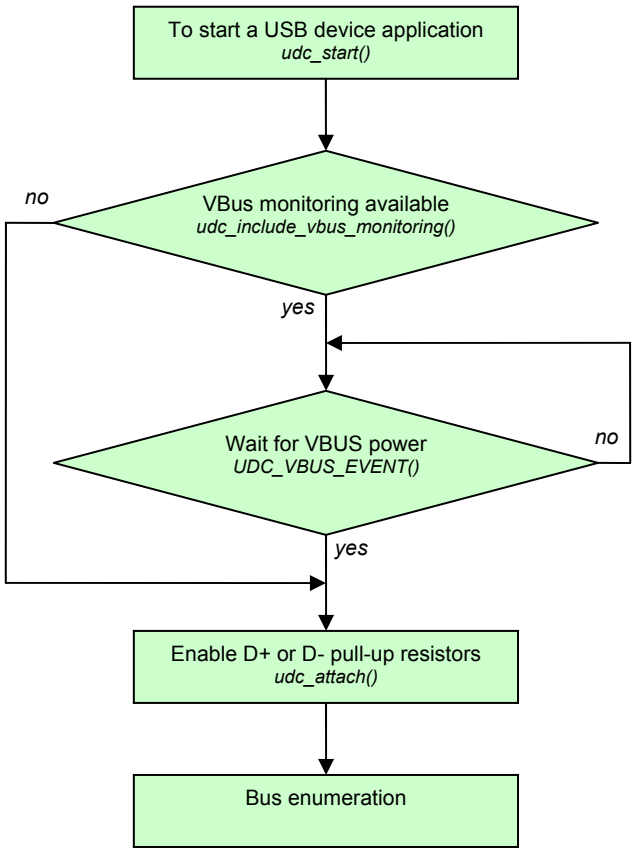
The implementation is made of three steps:

1. Start USB device.
2. Wait the enable of HID keyboard interface by the Host.
3. Scan the keys of the keyboard and send USB keyboard events if any.

6.3.1 USB device control

Only two function calls are needed to start a USB device application, see Figure 6-1.

Figure 6-1. USB device application sequence.



NOTE

In case of a new project, the USB stack requires to enable interrupts and to initialize the clock and sleepmgr services.

Example:

```
<conf_usb.h>
#define UDC_VBUS_EVENT(b_vbus_high) \
       vbus_event(b_vbus_high)

<main C file>:
main() {
   // Authorize interrupts
   irq_initialize_vectors();
   cpu_irq_enable();
   // Initialize the sleep manager service
   sleepmgr_init();
   // Initialize the clock service
   sysclk_init();
   // Enable USB Stack Device
   udc_start();
}

vbus_event(b_vbus_high) {
   if (b_vbus_high) {
       // Connect USB device
       udc_attach();
   }else{
       // Disconnect USB device
       udc_detach();
   }
}
```

6.3.2 USB interface control

After the device enumeration (detecting and identifying USB devices), the USB Host starts the device configuration. When the USB keyboard interface from device is accepted, the USB host enables this interface and the UDI_HID_KBD_ENABLE_EXT() callback function is called.

When the USB device is unplugged or is reset by the USB Host, the USB interface is disabled and the UDI_HID_KBD_DISABLE_EXT() callback function is called.

Thus, it is recommended to enable/disable sensors used by keyboard in these functions.

Example:

```
<conf_usb.h>
#define UDI_HID_KBD_ENABLE_EXT() \
       kbd_enable()
#define UDI_HID_KBD_DISABLE_EXT() \
       kbd_disable()

<main C file>:
kbd_enable() {
   // Enable process which scans the keys
   …
   return true;
}
kbd_disable() {
   // Disable process which scans the keys
   …
}
```

6.3.3 USB keyboard control

The USB HID keyboard functions described in Table 6-1 allow the application to control the keyboard.

Table 6-1. UDI HID keyboard - functions.

Declaration	Description
udi_hid_kbd_modifier_up(uint8_t modifier_id)	Send events key modifier released
udi_hid_kbd_modifier_down(uint8_t modifier_id)	Send events key modifier pressed
udi_hid_kbd_up(uint8_t key_id)	Send events key released
udi_hid_kbd_down(uint8_t key_id)	Send events key pressed

Example:

```
process_to_scan_keys() {
   if (is_key_shift_is_released())
       udi_hid_kbd_modifier_down(HID_MODIFIER_LEFT_SHIFT);
   if (is_key_a_is_press())
       udi_hid_kbd_up(HID_A);
   ...
}
```

The USB HID keyboard callback UDI_HID_KBD_CHANGE_LED(uint8_t value) is called when the caps or numeric lock change.

When the value argument contains the flag HID_LED_NUM_LOCK, the numeric lock is enabled.

When the value argument contains the flag HID_LED_CAPS_LOCK, the caps lock is enabled.

Example:

```
<conf_usb.h file>:
#define UDI_HID_KBD_CHANGE_LED(value) ui_kbd_led(value)

<C file>:
void ui_kbd_led(uint8_t value)
{
       if (value & HID_LED_NUM_LOCK) {
              // Here, turn on Num LED
       } else{
              // Here, turn off Num LED
       }
       if (value & HID_LED_CAPS_LOCK) {
              // Here, turn on Caps LED
       } else {
              // Here, turn off Caps LED
       }
}
```

7 Keyboard in a USB composite device

The information required to build a composite device is available in the Atmel AVR4902 ASF - USB Composite Device application note. A familiarity with this application note is mandatory.

This section introduced only the specific information required to build a composite device with a HID keyboard interface.

7.1 USB configuration

In addition to the USB configuration described in Section 6.2, the following values must be defined in the `conf_usb.h` file:

USB_DEVICE_EP_CTRL_SIZE
Endpoint control size.

This must be:

- 8 for low speed device
- 8, 16, 32, or 64 for full speed device (8 is recommended to save RAM)
- 64 for a high speed device

UDI_HID_KBD_EP_IN
IN endpoint number used by the HID keyboard interface.

UDI_HID_KBD_IFACE_NUMBER
Interface number of the HID keyboard interface.

USB_DEVICE_MAX_EP
Total number of endpoints in the application. This must include one endpoint for HID keyboard interface.

7.2 USB descriptor

The USB device Descriptor of composite device, defined in the `conf_usb.h` file, must include the HID keyboard interface descriptor:

```
//! Define structure of composite interfaces descriptor
#define    UDI_COMPOSITE_DESC_T \
    udi_hid_kbd_desc_t udi_hid_kbd; \
    ...

//! Fill composite interfaces descriptor for Full Speed
#define    UDI_COMPOSITE_DESC_FS \
    .udi_hid_kbd             = UDI_HID_KBD_DESC, \
    ...

//! Fill composite interfaces descriptor for High Speed
#define    UDI_COMPOSITE_DESC_HS \
    .udi_hid_kbd             = UDI_HID_KBD_DESC, \
    ...

//! Fill Interface APIs corresponding at interfaces descriptor
#define    UDI_COMPOSITE_API  \
    &udi_api_hid_kbd,       \
    ...
```

8 Table of contents

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

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Milennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan
16F, Shin Osaki Kangyo Bldg.
1-6-4 Osaki Shinagawa-ku
Tokyo 104-0032
JAPAN
Tel: (+81) 3-6417-0300
Fax: (+81) 3-6417-0370

© 2011 Atmel Corporation. All rights reserved.

Atmel®, Atmel logo and combinations thereof, AVR®, AVR Studio®, XMEGA®, and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Windows® and others are registered trademarks or trademarks of Microsoft Corporation in U.S. and or other countries. Other terms and product names may be trademarks of others.

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 THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, 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 AND 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 products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

8446A-AVR-10/11