Atmel AVR4905: ASF - USB Device HID Generic

Features

- **USB 2.0 compliance**
  - **Chapter 9 compliance**
  - **Low Speed (1.5Mbit/s), Full Speed (12Mbit/s), High Speed (480Mbit/s) data rates**
- **IN and OUT interrupt transfer**
- **Small stack size frees space for main application**
- **Remote wakeup support**
- **USB bus powered support**
- **Real time (O.S. compliance, interrupt driven)**
- **Support 8-bit and 32-bit Atmel® AVR®**

1 Introduction

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

8-/32-bit Atmel Microcontrollers

Application Note

Rev. 8499A-AVR-03/12

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

3 Overview

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

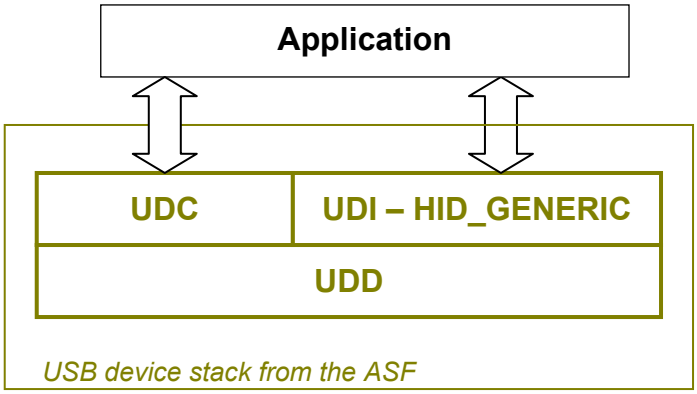
- **HID Generic Solution Introduction**
  Provides information about the HID Generic solution's advantages
- **Quick start**
  Describes how to start a ready to use HID Generic device example
- **Demo behavior**
  Describes the entire demo, PC application and device
- **Example description**
  Describes a HID Generic device example
- **Building a USB device HID Generic**
  Describes how to add a USB HID Generic device interface in a project
- **HID Generic in a USB composite device**
  Describes how to integrate a HID Generic interface in a composite device project

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

- User application
- USB device Interface HID Generic (UDI - HID_GENERIC)
- USB device Controller (UDC)
- USB device Driver (UDD)

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

Figure 3-1. USB HID Generic class solution architecture.



NOTE The USB device stack is available in the ASF in the common/services/usb directory.

4 HID Generic solution introduction

The USB interface becomes very complex when the user data does not fit with USB standard classes (Mass Storage, Audio, Video...). Specific drivers must be developed, requiring a significant amount of development time.

Atmel has developed a solution to save time and development efforts. This solution is based on HID class. It ensures a full duplex transfer between the device and the PC (up to 64Kbytes/s for Full Speed devices).

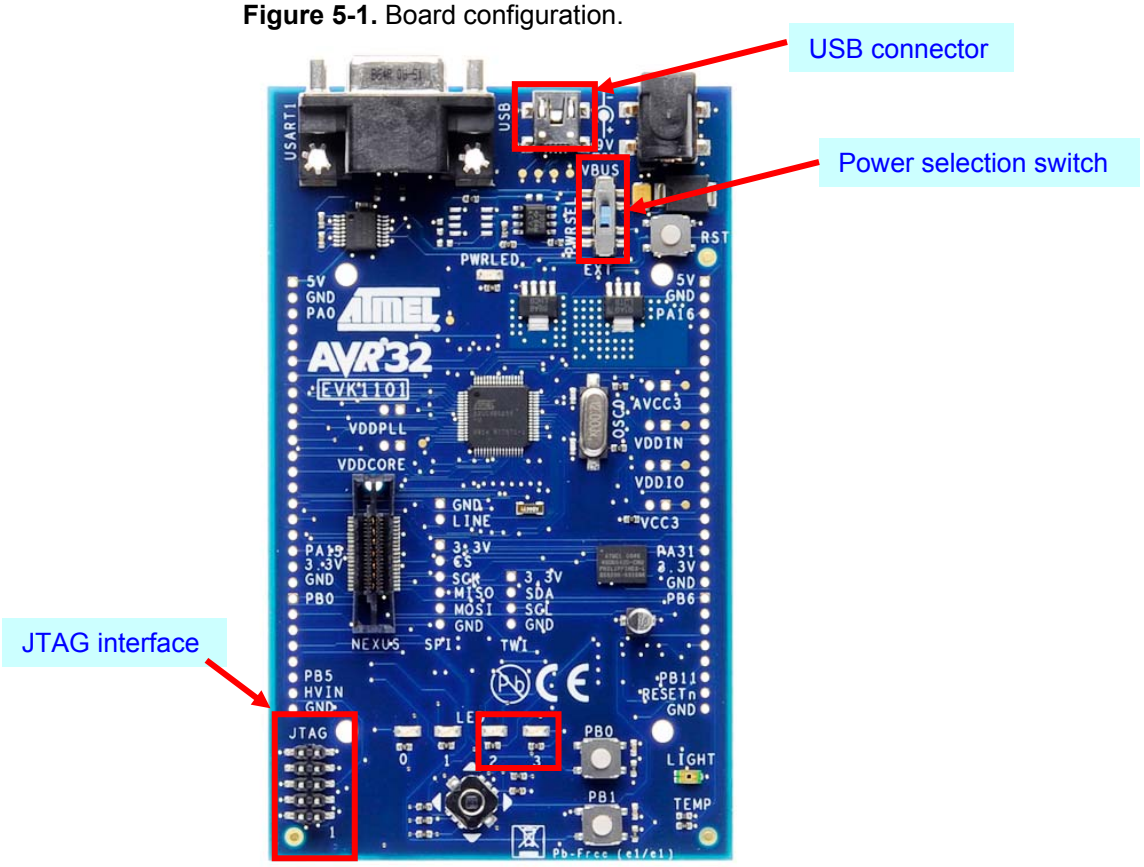
The HID class is supported by all Microsoft® OS from Windows® 98SE and later. It is also supported by most other OS running on PCs (at time of publication).

5 Quick start

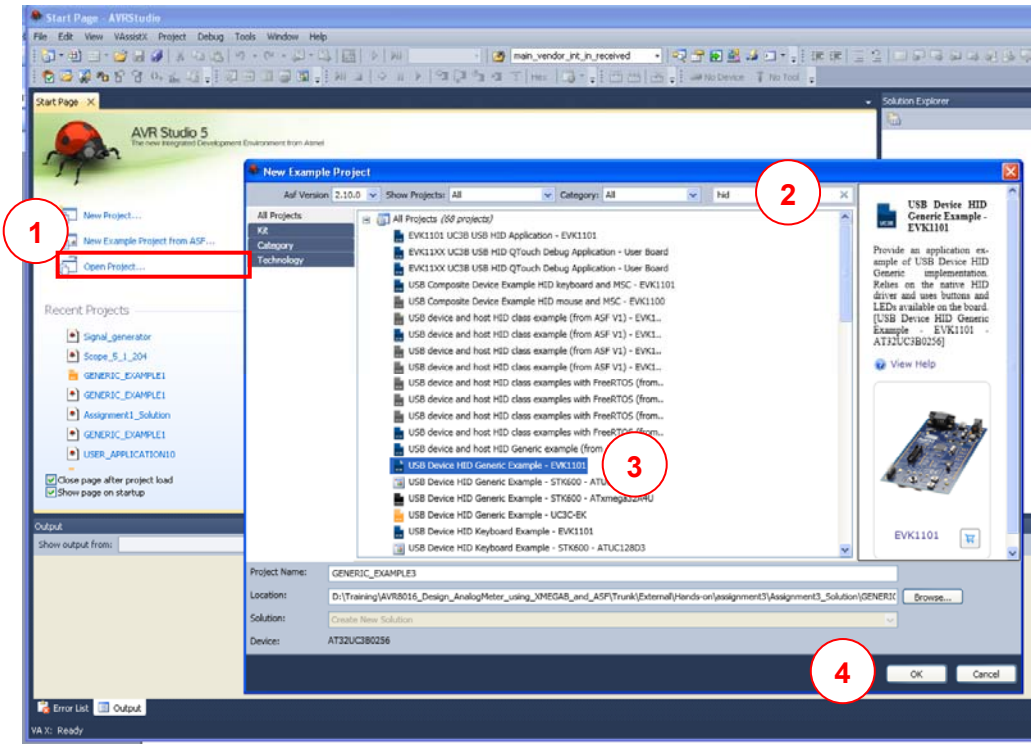
The USB device HID Generic examples are available in Atmel AVR Studio® 5 and ASF. Hereunder is the procedure to start a USB device HID Generic class example quickly:

1. Connect the board (for example the Atmel EVK1101 kit) to the PC using the USB cable.
2. Connect the Atmel debugger (hereafter the JTAGICE 3) to program the device.
3. Power selection switch must be set to the VBUS position.

Figure 5-1. Board configuration.



4. Start Atmel AVR Studio 5 or later and click on “*New Example Project from ASF*”. In the New Example Project’s list, select “USB Device HID Generic Example” based on the board you are using. The filter list can be used to find the example quickly (that is, use the HID keyword).



5. 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.

NOTE

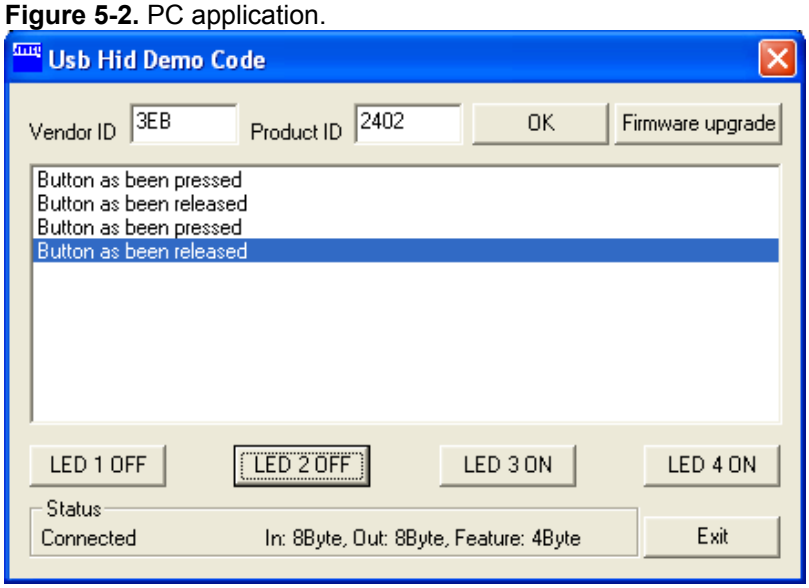
Since the HID Generic is using the native driver, the device install is silent.

5.1 Demo behavior

Now, since the device is enumerated, you can start the demo:

1. A PC application is available in: http://www.atmel.com/dyn/resources/prod_documents/AVR153.zip. Download and unzip the file.
2. From the “ExeDemo” folder launch the “UsbHidDemoCode.exe” application.
3. Modify the PID to 2402 as shown below and click on the ‘OK’ button.

Figure 5-2. PC application.



1. Use the LED 2 ON/OFF and LED 3 ON/OFF buttons to turn ON and OFF the LED 2 and 3 of the EVK1101 (if you are using different board, please refer to the ASF Example Project Help).
2. Push and release the buttons PB0 and PB1 to see the status displayed on the PC application window.

NOTE The Firmware upgrade button is used to disconnect the device from the USB bus

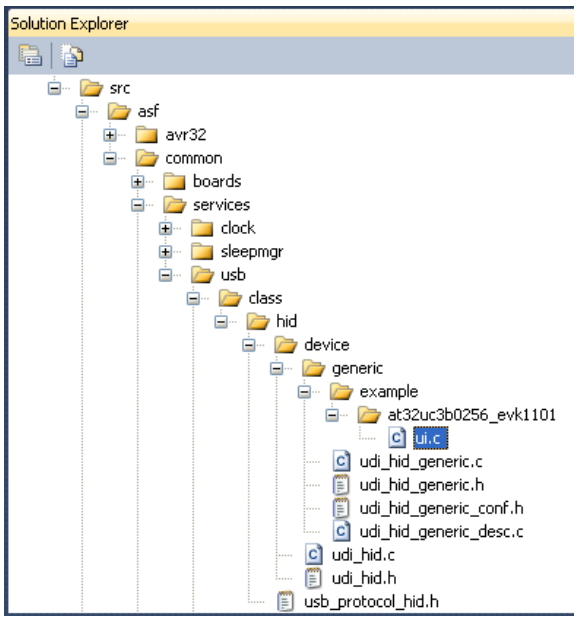
5.1.1 Hardware behavior

During the application execution the hardware (used board) should behave as below (LED number depends on the used board):

- A LED is on when USB device is in active mode and is off in SUSPEND mode
- A LED blinks showing that HID Generic interface is enabled by the USB Host
- LEDs are switched ON/OFF using the PC application.

NOTE

The user interface description (specific to the board) is defined at the end of `ui.c` source file.



6 Example description

6.1 Example content

The ASF provides a USB device HID Generic class example for various Atmel AVR products. All these examples share common files and implement the same application.

The Table 6-1 introduces a summary of the main files included in the USB device HID Generic example. These files are associated to the modules described in Figure 3-1.

Table 6-1. USB device HID Generic example files.

Modules	Files	ASF paths	Description

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

UDI GENERIC	`udi_hid_generic.c/h`	common/services/usb/class/hid/device/generic/	HID Generic implementation
	`udi_hid_genericdesc.c` `udi_hid_generic_conf.h`	common/services/usb/class/hid/device/generic/	USB Descriptors for an USB device with HID Generic interface (not applicable for USB composite device)

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

6.2 Example behavior

The `main.c` and `ui.c` files implement the user interface HID Generic application. It is based on three 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 Generic interface via call-back and enable the loopback process:

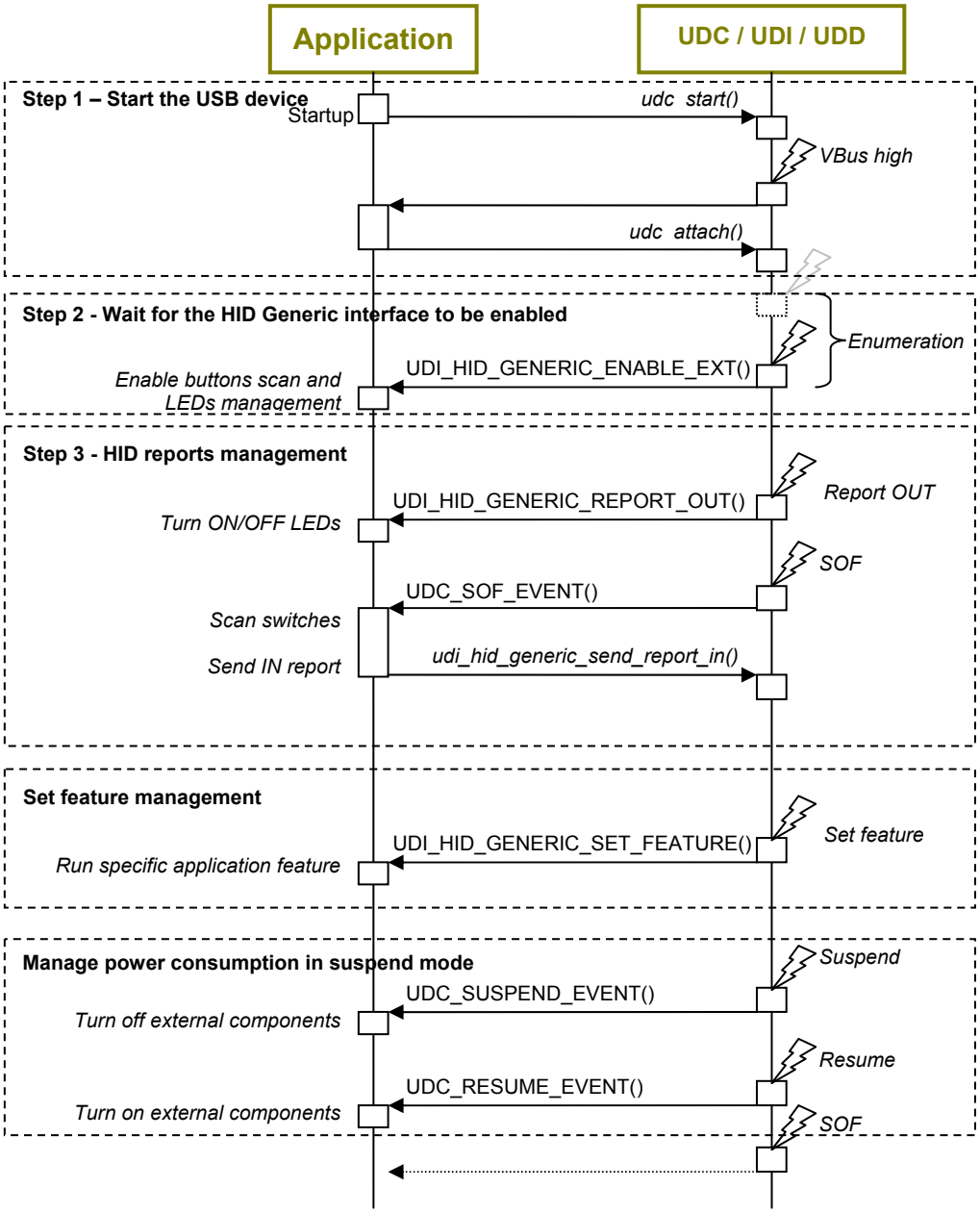
```
UDI_HID_GENERIC_ENABLE_EXT() // Interface enabled callback
```

3. Scan buttons and manage LEDs according to the host request:

```
//When button 0 changes status, send the report to the host
if (b_btn_state != btn0_last_state) {
            ui_hid_report[0] = b_btn_state;
            udi_hid_generic_send_report_in(ui_hid_report);
            btn0_last_state = b_btn_state;
    }

//Turn ON/OFF the LEDs according to the host report
UDI_HID_GENERIC_SET_FEATURE(report)
{
if (report[0]=='1') {
            // A led must be on
            switch(report[1]) {
                    case '2':
                    LED_On(LED2);
                    break;
                    case '3':
                    LED_On(LED3);
                    break;
            }
}
```

Figure 6-1. Example behavior sequence.



7 Building a USB device HID Generic

The USB device HID Generic modules provide a USB HID Generic interface which can be used to build a USB application.

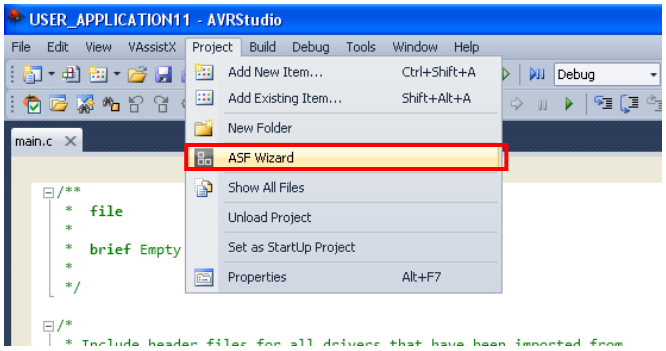
These modules are available in Atmel AVR Studio 5 and can be imported in an AVR Studio 5 project. This section describes how to add a USB device HID Generic in a project:

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

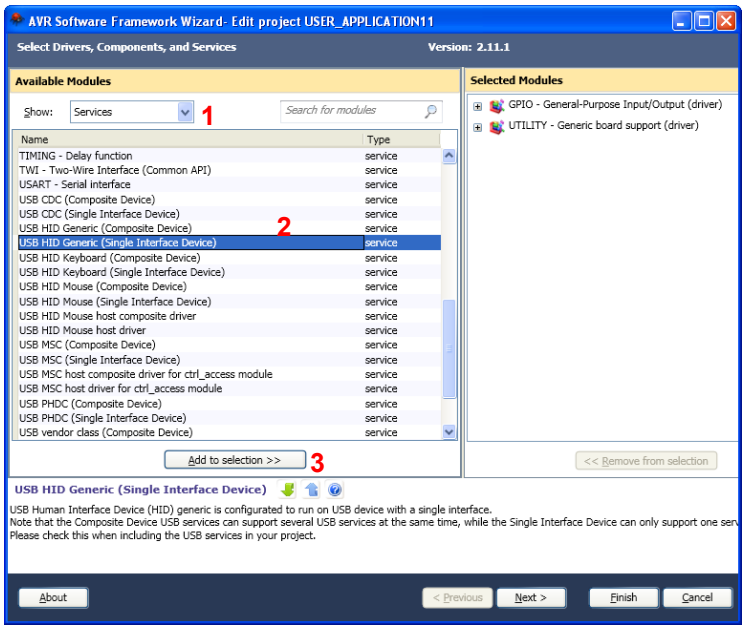
7.1 Import USB module

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

1. Open or create your project.
2. From the Project menu, choose “ASF Wizard.”



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



7.2 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 Atmel AVR4900: ASF – USB Device Stack application note in the Section 7.1.1: USB device configuration".

The UDD configuration possibilities are described in the Atmel AVR4900: ASF – USB Device Stack application note in the Section 7.1.3: USB drivers' configuration".

The UDI which is the HID Generic interface require some configuration described in Table 7-1.

Table 7-1. UDI HID Generic – configuration.

Define name	Type	Description
UDI_HID_GENERIC_ENABLE_EXT	Call-back function	Call-back function called when HID Generic interface is enabled
UDI_HID_GENERIC_DISABLE_EXT	Call-back function	Call-back function called when HID Generic interface is disabled
UDI_HID_GENERIC_REPORT_OUT	Call-back function	Call-back function called when OUT report request is received
UDI_HID_GENERIC_SET_FEATURE	Call-back function	Call-back function called when a set feature request is received
UDI_HID_REPORT_IN_SIZE UDI_HID_REPORT_OUT_SIZE UDI_HID_REPORT_FEATURE_SIZE	Constant	Report IN size (send through the EP IN) Report OUT size (send through the EP OUT) Set Feature report size (send through the endpoint 0)

NOTE

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

7.3 USB implementation

This section describes source code to add to run a USB device HID Generic application.

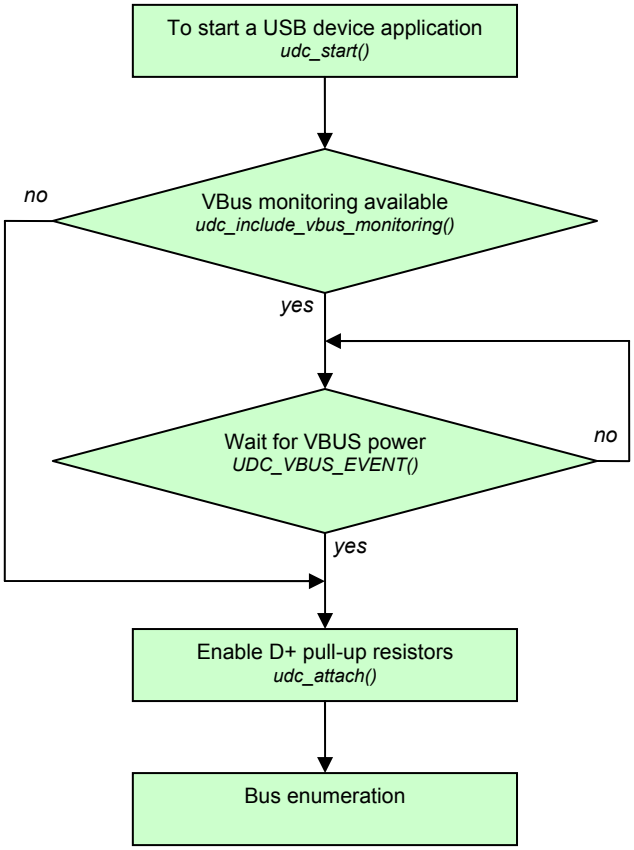
The implementation is made of three steps:

1. Start USB device.
2. Wait the enable of HID Generic interface by the Host.
3. Transfer data on USB bus.

7.3.1 USB device control

Only two function calls are needed to start a USB device application, see Figure 7-1.

Figure 7-1. 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();
   if (!udc_include_vbus_monitoring()) {
       // VBUS monitoring is not available on this product
```

```
        // thereby VBUS has to be considered as present
        vbus_event (true);
    }
}

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

7.3.2 USB interface control

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

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

Example:

```
<conf_usb.h>
#define UDI_HID_GENERIC_ENABLE_EXT() \
        Hid_generic_enable()
#define UDI_HID_GENERIC_DISABLE_EXT() \
        Hid_generic_disable()

<user C file>:
Hid_generic_enable() {
    // Start an ADC conversion
    …
    return true;
}
Hid_generic_disable() {
    // Stop the ADC conversion
    …
}
```

7.3.3 USB HID Generic functions

The USB HID Generic class functions described in Table 7-2 allows to send or to receive data.

Table 7-2. UDI HID Generic – data functions.

Declaration	Description
udi_hid_generic_send_report_in()	Sent a report to the host
#define UDI_HID_GENERIC_REPORT_OUT(report)	Get the report data sent by the host
#define UDI_HID_GENERIC_SET_FEATURE(report)	Get the Feature report data sent by the host

7.4 HID Generic and USB Host support

Even if the HID class is natively supported by Windows, an interface has to be created to interface with the drivers. To reduce the time of development, Atmel offers a DLL which provides functions to communicate with the HID Generic device.

For further information regarding this DLL and how to use it to create a PC application, please refer to the application note “USB PC Drivers Based on Generic HID Class” available via this link:

http://www.atmel.com/Images/doc7645.pdf

8 HID Generic class 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 Generic class interface.

8.1 USB configuration

In addition to the USB configuration described in Section 7.2, the following values must be defined in the `conf_usb.h` file:

USB_DEVICE_EP_CTRL_SIZE
Endpoint control size.

This must be:

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

UDI_HID_GENERIC_EP_IN
IN interrupt endpoint number used by the HID Generic interface.

UDI_HID_GENERIC_EP_OUT
OUT interrupt endpoint number used by the HID Generic interface.

UDI_HID_GENERIC_IFACE_NUMBER
Interface number of the HID Generic interface.

USB_DEVICE_MAX_EP
Total number of endpoints in the application. This must include the number of endpoints used by HID Generic interface.

8.2 USB descriptor

The USB device Descriptor of composite device, defined in `conf_usb.h` file, must include a HID Generic interface:

```
//! Define structure of composite interfaces descriptor
#define    UDI_COMPOSITE_DESC_T                    \
    udi_hid_generic_desc_t udi_hid_generic;              \
    ...

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

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

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

9 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

8499A-AVR-03/12