

Atmel AVR4901: ASF - USB Vendor Class Application



Features

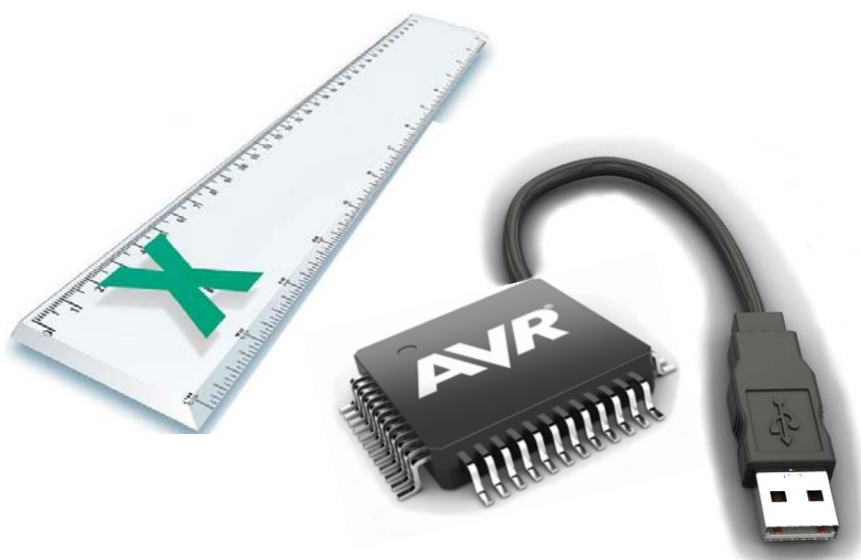
- USB 2.0 compliance
 - Chapter 9 compliance
 - Full Speed (12Mbit/s), High Speed (480Mbit/s) data rates
- Control, Bulk, Isochronous and Interrupt transfer types
- Small stack size frees space for main application
- Remote wakeup support
- USB bus powered support
- Real time (O.S. compliance, interrupt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

**8-/32-bit Atmel
Microcontrollers**

Application Note

1 Introduction

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



Rev. 8481A-AVR-01/12





2 Abbreviations

ASF:	AVR Software Framework
CD:	Composite Device: a USB device with more than one interface
CDC:	Communication Device Class
FS:	USB Full Speed
HID:	Human Interface Device
HS:	USB High Speed
LS:	USB Low Speed
MSC:	Mass Storage Class
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 six sections for all types of requirements when building a USB device vendor application:

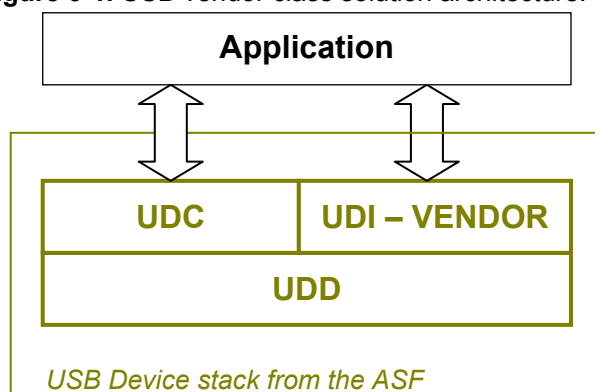
- **Vendor class Specification**
Provides information to help users with the vendor class specification and when they need such solution
- **Quick start**
Describes how to start a ready to use vendor class device example
- **Example description**
Describes a vendor class device example
- **Building a USB device vendor**
Describes how to add a USB device vendor interface in a project
- **Building a USB vendor class application**
Describes how to add a USB vendor class application
- **Vendor class in a USB composite device**
Describes how to integrate a vendor class interface in a composite device project

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

- User Application
- USB device Interface VENDOR (UDI-VENDOR)
- 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 vendor class solution architecture.



NOTE

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

4 Vendor class specification

The development of a vendor class can be required when the existing class (http://www.usb.org/developers/defined_class) does not correspond to the user specification.

Before starting a vendor class specification, check in [Table 4-1](#) that the native USB classes are not applicable for your USB application.

Table 4-1. Analysis of native classes.

Class	PROS.	CONS.
HID	- Native driver - USB Low Speed mode support - Suitable for controlling or monitoring an application	- Interrupt endpoint: - low and full speed 64KB/s - high speed 2.93MB/s
CDC	- High transfer rate - Simple Host application (terminal) - Easy migration for USART based applications	- Need of *.inf file under Windows® - Driver not WHQL certified
MSC	Native driver	SCSI protocol adapted only for mass storage
AUDIO	- Native driver - Interesting to record or send data stream	No handshake data transfer (isochronous)

When building a vendor class, the main important thing to specify is the transfer modes used by endpoints.

These can be chosen based on the description given in [Table 4-2](#).

Table 4-2. Endpoint type description.

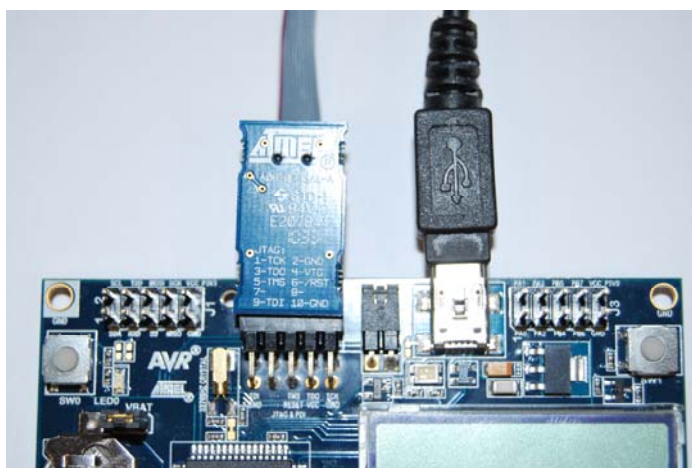
Endpoint type	Description	Use case
Control	- Low transfer rate - No need of new endpoint which can reduce application footprint	Configuration of the device
Interrupt	- Very low transfer - Frequency guaranteed	Real time constraints and low bandwidth transfer
Bulk	- High transfer rate - Transfer guaranteed (ACK)	Large amount of data transfer
Isochronous	- Very High transfer rate - Transfer not guaranteed	Streaming application

5 Quick start

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

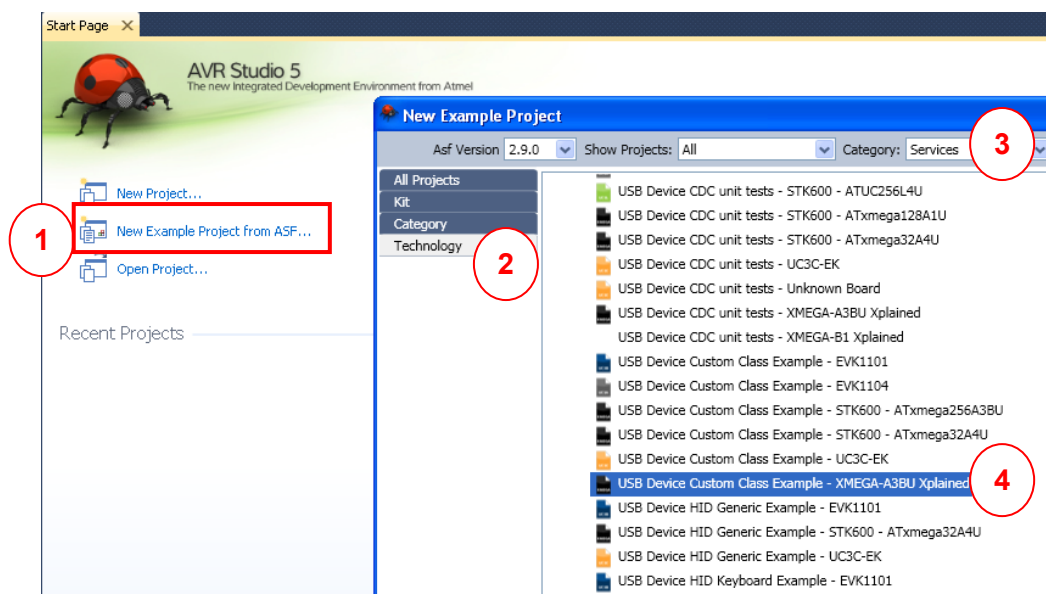
1. Connect the board (for example the Atmel AVR XMEGA®-A3BU Xplained kit) to the PC using the USB cable.
2. Connect the Atmel debugger (hereafter the JTAGICE3) to program the device.

Figure 5-1 Board connection.



3. Start AVR Studio 5 and click on “New Example Project from ASF”.

In the New Example Project’s list, select “USB Device Vendor Class Example” based on the board you are using. The filter list can be used to find the example quickly.

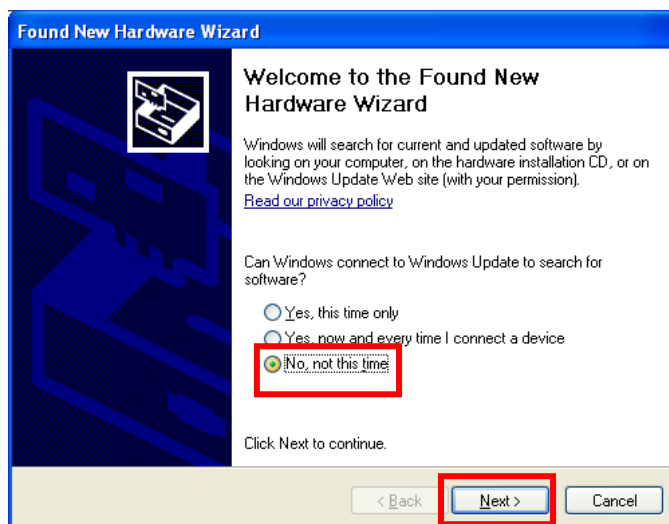




4. Compile, load and execute.

The project does not require any modification and only needs to be compiled, loaded and ran. Connect the Atmel debugger supported by the board and press F5.

When running the application for the first time a Found New Hardware Wizard pops up as showing below, select “No, not this time” and click on “Next”:

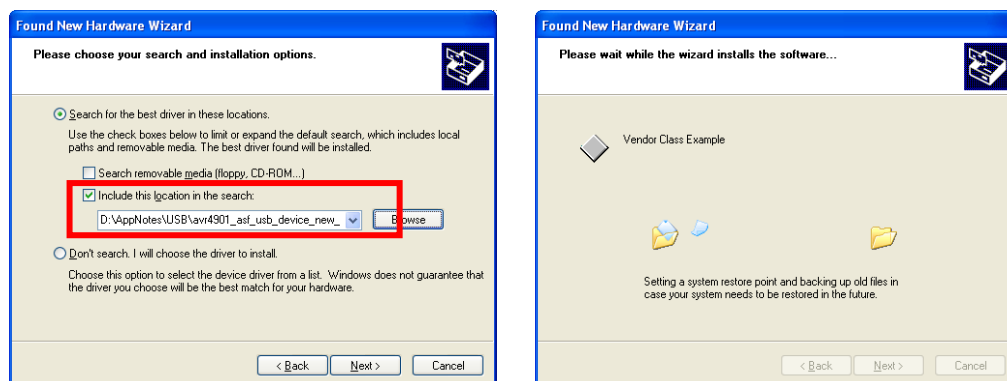


5. Load the driver using the inf file.

NOTE

Since the USB vendor class is not supported by any native driver, user needs to provide a proprietary driver or use a third party solution.

For this example, an open source solution “libusb” is used to support this device. An inf file was generated by the libusb tool. Point to the folder **avr4901/driver/** to install the driver.



Once the device is correctly installed, lunch the `avr_vendor_class_example.exe` from the folder `avr4901/tool/`. This program will perform a loopback in all endpoints. A status is displayed to show the correct execution of the application:

```

C:\windows\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\annaouar>D:\AppNotes\USB\avr4901_asf_usb_device_new_cl
ss\trunk\tool\avr_vendor_class_example.exe

----- PC tool for ASF vendor class example (V1.1)
Initialization library "libusb"...
Search device...
USB Device of ASF vendor class example found:
- Device version: 1.0
- Manufacture name: ATMEL ASF
- Product name: Vendor Class Example
Initialization device...
Control endpoint loop back...
Interrupt endpoint loop back...
Bulk endpoint loop back...
Isochronous endpoint loop back...
----- Tests completed.

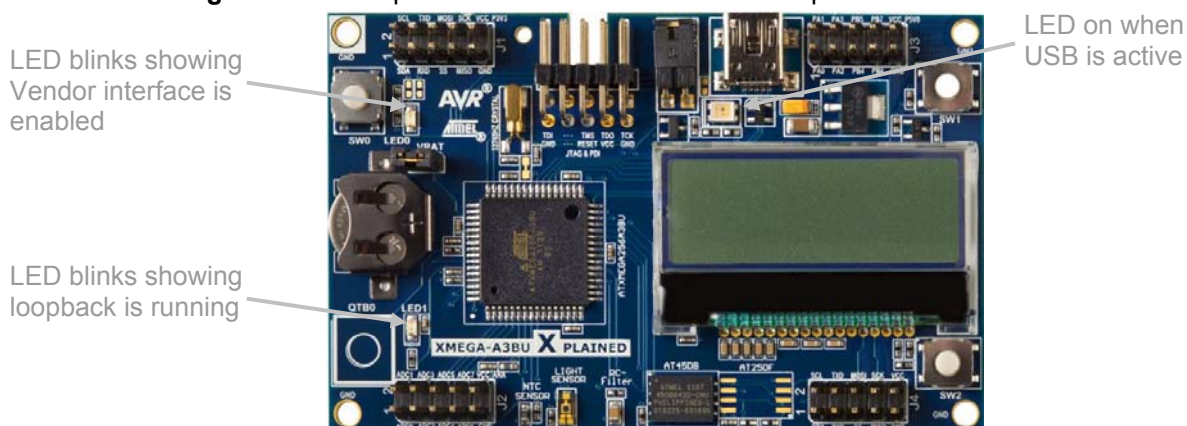
C:\Documents and Settings\annaouar>_
  
```

5.1 Hardware behavior

During the application execution the hardware (used board) should behave as below:

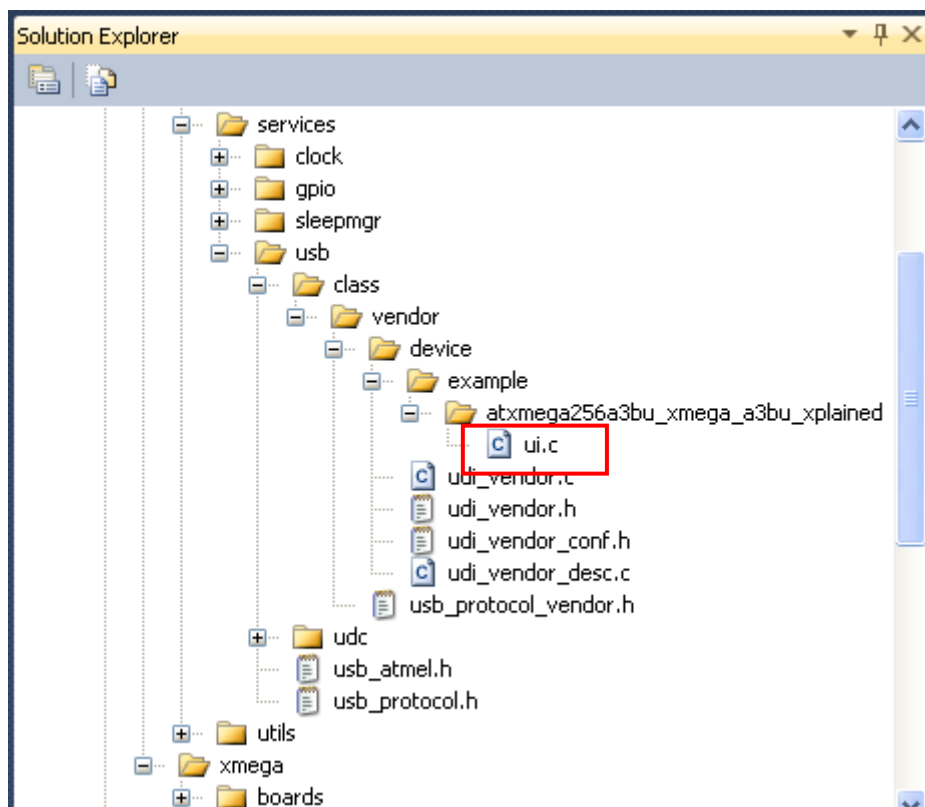
- A LED is on when USB device is in active mode and is off in SUSPEND mode
- A LED blinks showing that vendor class interface is enabled by the USB Host
- A LED is on when the loopback is running

Figure 5-2. Example with the Atmel XMEGA-A3BU Xplained board.



The user interface description (specific to the board) is defined at the end of `ui.c` source file. This file is available within the “Solution Explorer” under “common/services/usb/class/vendor/device/example/part_number_board/”.

Figure 5-3 `ui.c` file location.



6 Example description

6.1 Example content

The ASF provides a USB device vendor 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 vendor example. These files are associated to the modules described in [Figure 3-1](#).

Table 6-1. USB device VENDOR 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 VENDOR	udi_vendor.c/h	common/services/usb/class/vendor/device/	Vendor class implementation
	udi_vendor_desc.c udi_vendor_conf.h	common/services/usb/class/vendor/device/	USB Descriptors for an USB device with vendor 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 vendor 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 vendor interface via call-back and enable the loopback process:

```
UDI_VENDOR_ENABLE_EXT() // Interface enabled callback
```

The callback calls:

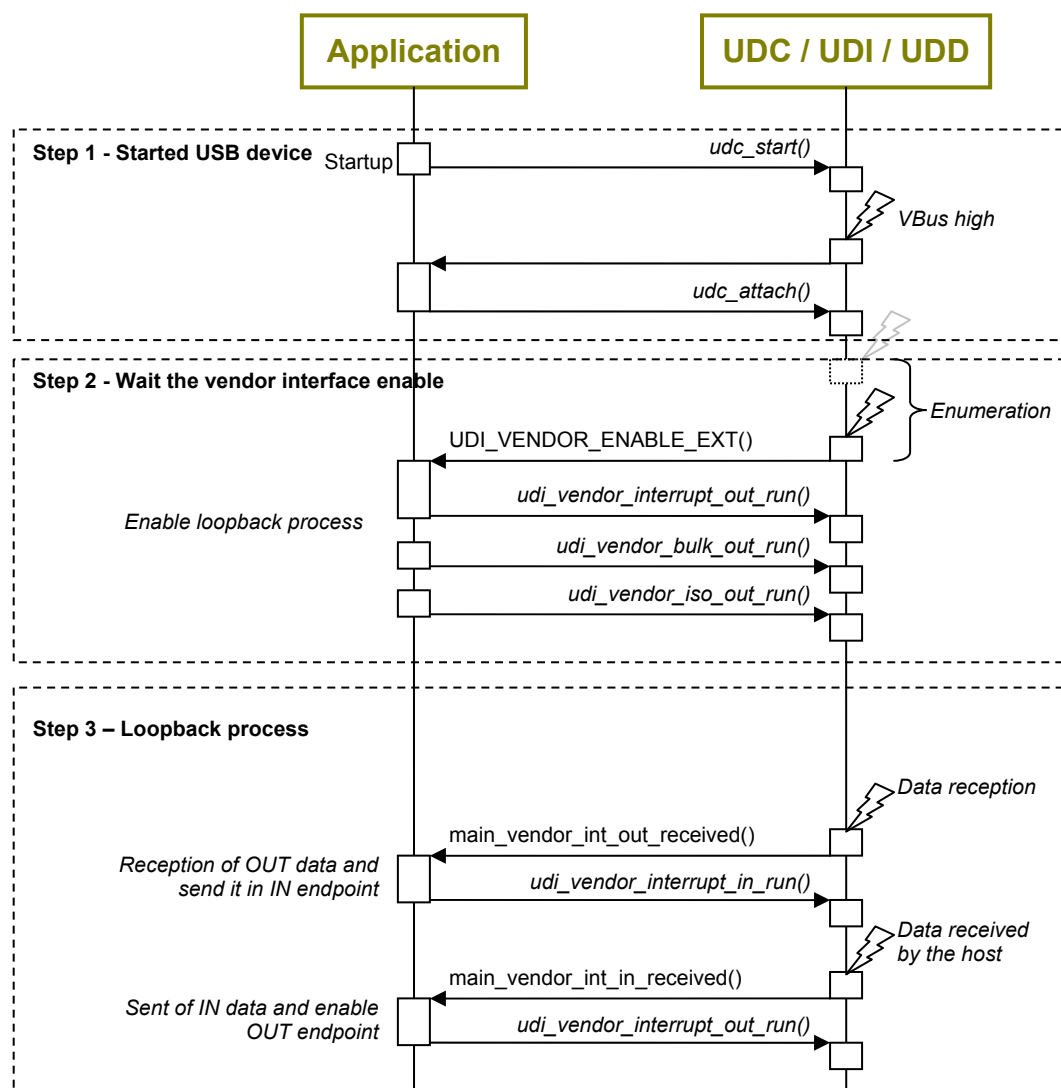
```
- udi_vendor_bulk_out_run() // Enable transfer on interrupt OUT endpoint  
- udi_vendor_interrupt_out_run() // Enable transfer on interrupt OUT endpoint  
- udi_vendor_iso_out_run() // Enable transfer on isochronous OUT endpoint
```

3. Perform loopback in all endpoints (Control, Interrupt, Bulk and Isochronous):

```
// When a OUT data is received, then send this data on IN
void main_vendor_int_out_received(udd_ep_status_t status,
    iram_size_t nb_transferred)
{
    if (UDD_EP_TRANSFER_OK != status) {
        return; // Tranfert aborted, then stop loopback
    }
    ui_loop_back_state(true);
    // Send on IN endpoint the data received on endpoint OUT
    udi_vendor_interrupt_in_run(
        main_buf_loopback,
        nb_transferred,
        main_vendor_int_in_received);
}

// When a IN data has been sent, then the OUT transfer is enabled again
void main_vendor_int_in_received(udd_ep_status_t status,
    iram_size_t nb_transferred)
{
    if (UDD_EP_TRANSFER_OK != status) {
        return; // Tranfert aborted, then stop loopback
    }
    ui_loop_back_state(false);
    // Wait a full buffer
    udi_vendor_interrupt_out_run(
        main_buf_loopback,
        sizeof(main_buf_loopback),
        main_vendor_int_out_received);
}
```

Figure 6-1. Example behavior sequence.



7 Building a USB device vendor

The USB device vendor modules provide a USB vendor interface which can be used to build a USB vendor class application.

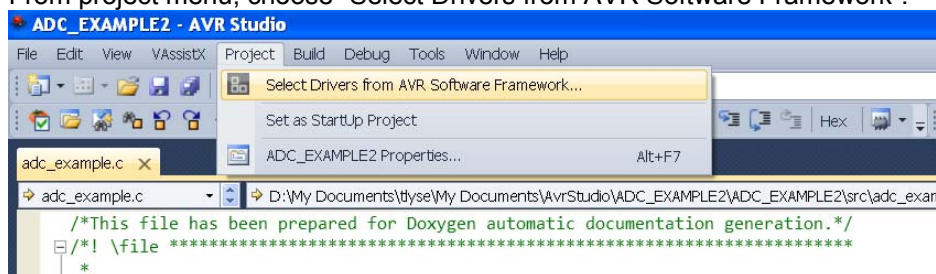
These modules are available in Atmel AVR Studio 5 and can be imported in an AVR Studio 5 project. This section describes how to add a USB device vendor in a project:

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

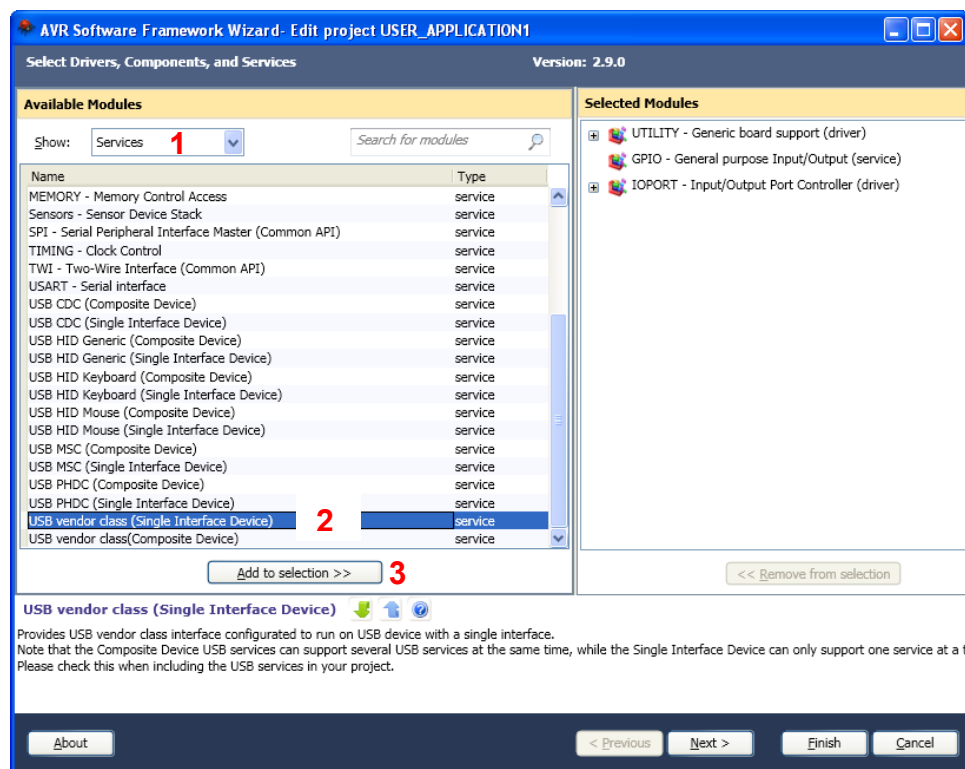
7.1 Import USB module

To import the USB vendor 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 vendor class (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 vendor interface require some configuration described in [Table 7-1](#).

Table 7-1. UDI vendor – configuration.

Define name	Type	Description
UDI_VENDOR_ENABLE_EXT	Call-back function	Call-back function called when vendor interface is enabled
UDI_VENDOR_DISABLE_EXT	Call-back function	Call-back function called when vendor interface is disabled
UDI_VENDOR_SETUP_OUT_RECEIVED	Call-back function	Call-back function called when OUT setup request is received
UDI_VENDOR_SETUP_IN_RECEIVED	Call-back function	Call-back function called when IN setup request is received
UDI_VENDOR_EPS_SIZE_INT_FS UDI_VENDOR_EPS_SIZE_BULK_FS UDI_VENDOR_EPS_SIZE_ISO_FS	constant	Interrupt endpoints size for full speed (up to 64) Bulk endpoints size for full speed (8, 16, 32 or 64) Isochronous size for full speed (up to 1023)
UDI_VENDOR_EPS_SIZE_INT_HS UDI_VENDOR_EPS_SIZE_BULK_HS UDI_VENDOR_EPS_SIZE_ISO_HS	constant	Interrupt endpoints size for full speed (up to 1024) Bulk endpoints size for full speed (8, 16, 32,...,512) Isochronous size for full speed (up to 1024)

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 vendor application.

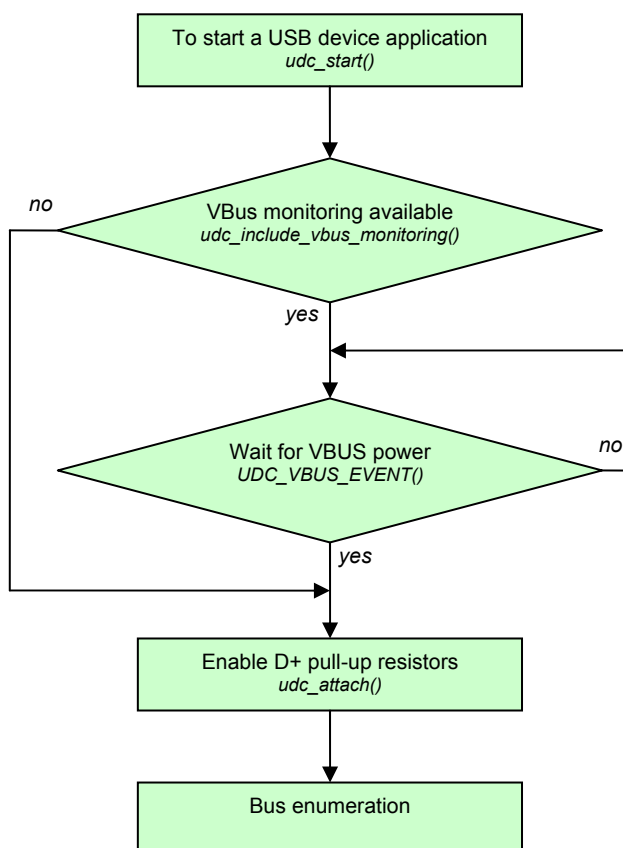
The implementation is made of three steps:

1. Start USB device.
2. Wait the enable of vendor 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 vendor class interface is accepted, the USB host enables this interface and the UDI_VENDOR_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_VENDOR_DISABLE_EXT() callback function is called.

Example:

```
<conf_usb.h>
#define UDI_VENDOR_ENABLE_EXT() \
    vendor_enable()
#define UDI_VENDOR_DISABLE_EXT() \
    vendor_disable()

<user C file>:
vendor_enable() {
    // Start an ADC conversion
    ...
    return true;
}
vendor_disable() {
    // Stop the ADC conversion
    ...
}
```

7.3.3 USB vendor functions

The USB vendor class functions described in Table 7-2 allow to send or to receive data.

Table 7-2. UDI vendor class – data functions.

Declaration	Description
udi_vendor_interrupt_in_run ()	Start transfer in an IN interrupt endpoint
udi_vendor_interrupt_out_run ()	Start transfer in an OUT interrupt endpoint
udi_vendor_bulk_in_run ()	Start transfer in an IN bulk endpoint
udi_vendor_bulk_out_run ()	Start transfer in an OUT bulk endpoint
udi_vendor_iso_in_run ()	Start transfer in an IN isochronous endpoint
udi_vendor_iso_out_run ()	Start transfer in an OUT isochronous endpoint

7.4 Vendor class and USB Host support

Since the USB vendor class is specified by the user, there is no native support with any operating system. The user has to build an own driver and host application. However some solutions exist to help to build a fast solution. The most popular and free of charge are:

- Libusb: This solution has the advantage to support multi-operating system platforms (Windows, Linux®, Mac OS®) and offers an open source solution. Atmel provide an example using this solution available within the Atmel AVR4901 application note. For further information regarding libusb, please refer to <http://www.libusb.org/>
- WinUSB: This solution is provided by and dedicated to Windows platforms (native form Windows Vista®). WinUSB does not support isochronous transfer mode. For further information please refer to [http://msdn.microsoft.com/en-us/library/windows/hardware/ff540196\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/hardware/ff540196(v=vs.85).aspx)

8 Vendor 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 vendor 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_VENDOR_EP_INTERRUPT_IN

IN interrupt endpoint number used by the vendor interface (ignored if `UDI_VENDOR_EPS_SIZE_INT_FS` is 0).

UDI_VENDOR_EP_INTERRUPT_OUT

OUT interrupt endpoint number used by the vendor interface (ignored if `UDI_VENDOR_EPS_SIZE_INT_FS` is 0).

UDI_VENDOR_EP_BULK_IN

IN bulk endpoint number used by the vendor interface (ignored if `UDI_VENDOR_EPS_SIZE_BULK_FS` is 0).

UDI_VENDOR_EP_BULK_OUT

OUT bulk endpoint number used by the vendor interface (ignored if `UDI_VENDOR_EPS_SIZE_BULK_FS` is 0).

UDI_VENDOR_EP_ISO_IN

IN isochronous endpoint number used by the vendor interface (ignored if `UDI_VENDOR_EPS_SIZE_ISO_FS` is 0).

UDI_VENDOR_EP_ISO_OUT

OUT isochronous endpoint number used by the vendor interface (ignored if `UDI_VENDOR_EPS_SIZE_ISO_FS` is 0).

UDI_VENDOR_IFACE_NUMBER

Interface number of the vendor interface.

USB_DEVICE_MAX_EP

Total number of endpoints in the application. This must include the number of endpoints used by vendor interface (0 to 6 depend on `UDI_VENDOR_EPS_SIZE_X` defines).

8.2 USB descriptor

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

```
//! Define structure of composite interfaces descriptor
#define    UDI_COMPOSITE_DESC_T                \
    udi_vendor_desc_t udi_vendor;              \
    ...

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

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

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

9 Table of contents

<i>Atmel AVR4901: ASF - USB Vendor Class Application</i>	<i>1</i>
<i>Features</i>	<i>1</i>
<i>1 Introduction</i>	<i>1</i>
<i>2 Abbreviations</i>	<i>2</i>
<i>3 Overview</i>	<i>3</i>
<i>4 Vendor class specification</i>	<i>4</i>
<i>5 Quick start</i>	<i>5</i>
5.1 Hardware behavior	7
<i>6 Example description</i>	<i>9</i>
6.1 Example content	9
6.2 Example behavior	9
<i>7 Building a USB device vendor</i>	<i>12</i>
7.1 Import USB module	12
7.2 USB configuration	13
7.3 USB implementation	13
7.3.1 USB device control	13
7.3.2 USB interface control	15
7.3.3 USB vendor functions	16
7.4 Vendor class and USB Host support	16
<i>8 Vendor class in a USB composite device</i>	<i>17</i>
8.1 USB configuration	17
8.2 USB descriptor	18
<i>9 Table of contents</i>	<i>19</i>



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, Millennium 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

© 2012 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.