Atmel AVR4902: ASF - USB Composite Device

Features

- **USB 2.0 compliance**
  - **Chapter 9 certified**
  - **Control, Bulk, Isochronous and Interrupt transfer types**
  - **Low Speed (1.5Mbit/s), Full Speed (12Mbit/s), High Speed (480Mbit/s) data rates**
- **Small stack size free space for main application**
- **Real time (OS compliance, no latency)**
- **Supports 8-bit and 32-bit AVR® platforms**
- **USB DMA support increases speed performance**
- **Supports most USB classes and ready to use (HID, CDC, MSC, PHDC)**

1 Introduction

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

This document is associated to USB device class Atmel® application notes AVR4903 to AVR4909.

Atmel Microcontrollers

Application Note

Rev. 8445A-AVR-10/11

2 Abbreviations

ASF:	AVR Software Framework
CDC:	Communication Device Class
FS:	USB Full Speed
HID:	Human interface device
HS:	USB High Speed
UDC:	USB device Controller
UDD:	USB device Descriptor
UDI:	USB device Interface
USB:	Universal Serial Bus
MSC:	Mass Storage Class
PHDC:	Peripheral Health Device Class

3 Overview

This document includes two sections:

- **Quick start**
  Describes how to start a ready to use composite device example
- **Building a USB composite device**
  Describes how to create a USB composite device

For all these sections, it is recommended to know the main modules organization of a USB composite device application:

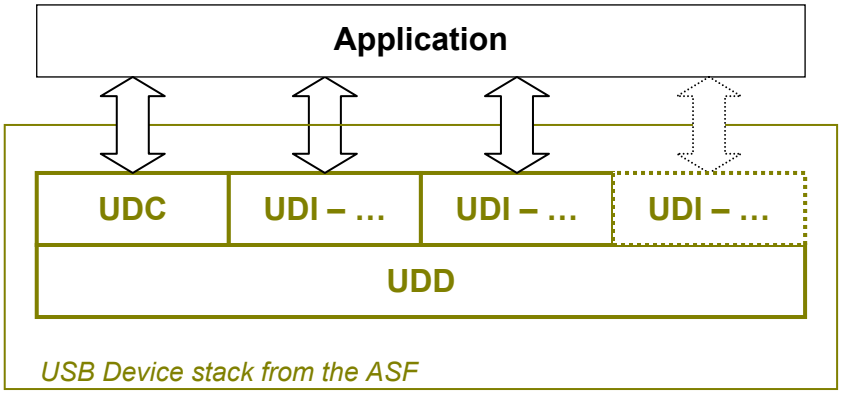
- User Application
- USB device Interfaces (UDI)
- USB device Controller (UDC)
- USB device Driver (UDD)

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

The current UDIs supported in a composite device are:

- USB device Mouse Interfaces (UDI-HID Mouse)
- USB device Keyboard Interfaces (UDI-HID Keyboard)
- USB device HID Generic Interfaces (UDI-HID Generic)
- USB device Mass Storage Interface (UDI-MSC)
- USB device Communication Interface (UDI-CDC)
- USB device Personal Health Interfaces (UDI-PHDC)

Figure 3-1. USB composite device architecture.

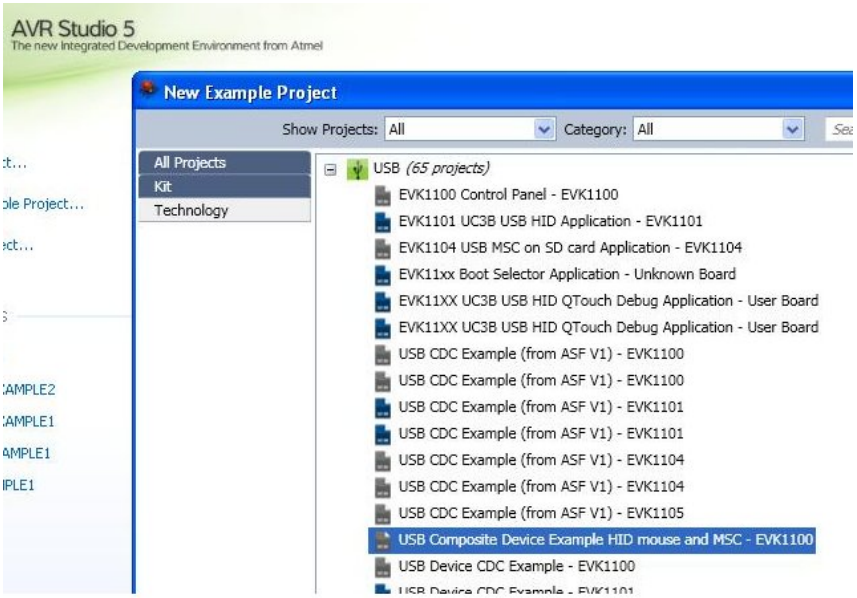


NOTE The USB device stack is available in the ASF in the common/services/usb/ directory.

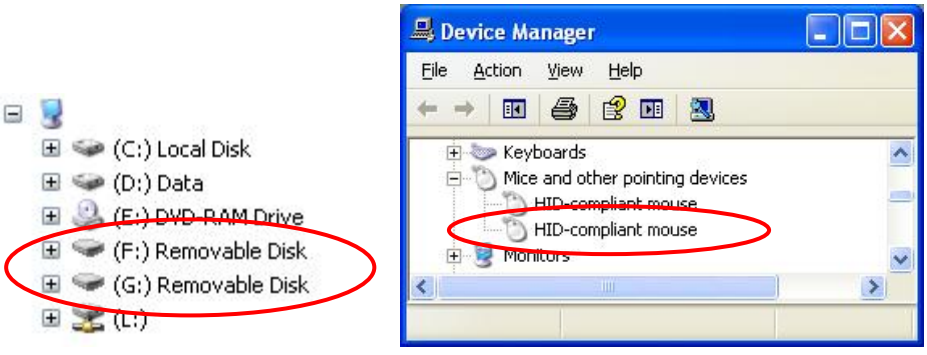
4 Quick start

The USB composite device examples are available in Atmel AVR Studio® 5 and ASF. Here, the examples provide a composite device supporting a mouse and U-Disk.

1. Powering the board through the USB connection.
   Connect a USB cable between board and 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 Composite Device Example HID mouse and MSC” 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.
4. Use it.
   One or more new removal disks are displayed on the USB Host and new mouse is installed.



The board user's interface to control the mouse and monitor the disk access is described at the end of `ui.c` source file provided in the example.

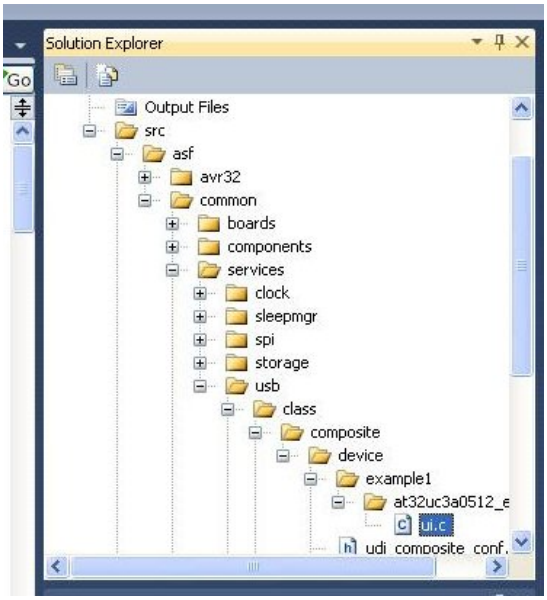
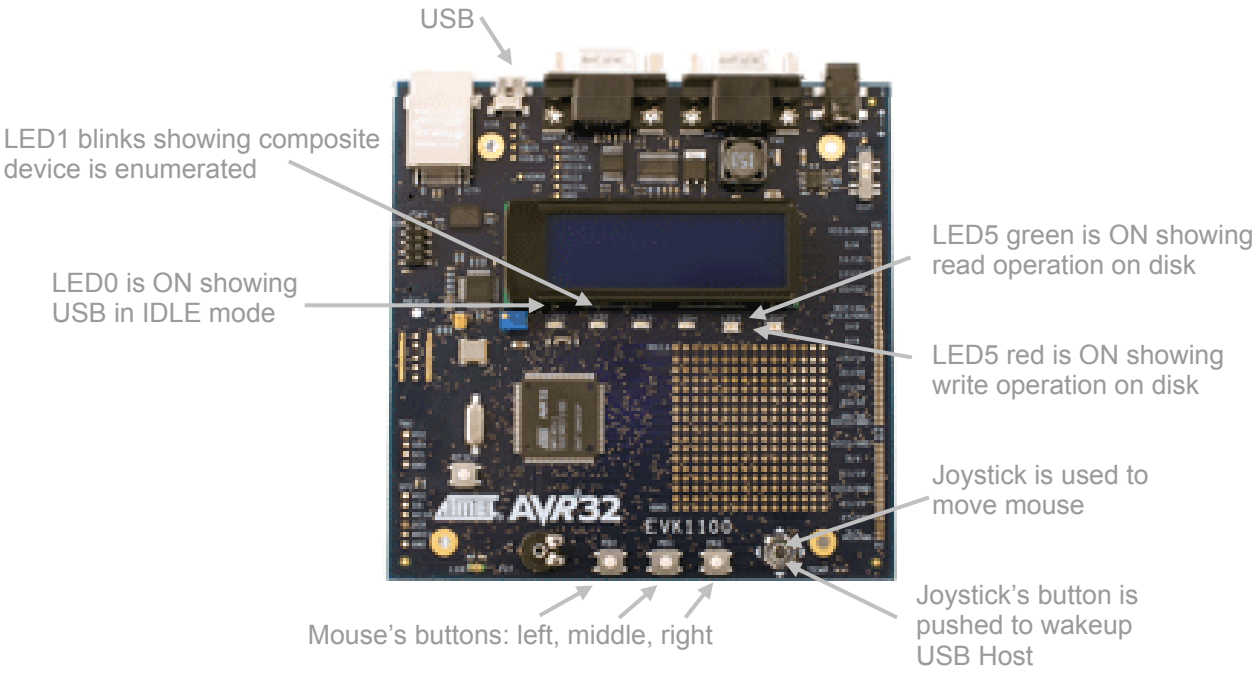


Figure 4-1. Example of user interface description using an Atmel EVK1100 board.



5 Building a USB composite device

The Atmel AVR Software Framework (ASF) provides many USB Interface modules which can be added on a USB composite device. These modules are available in Atmel AVR Studio 5 and can be imported in an AVR Studio 5 project.

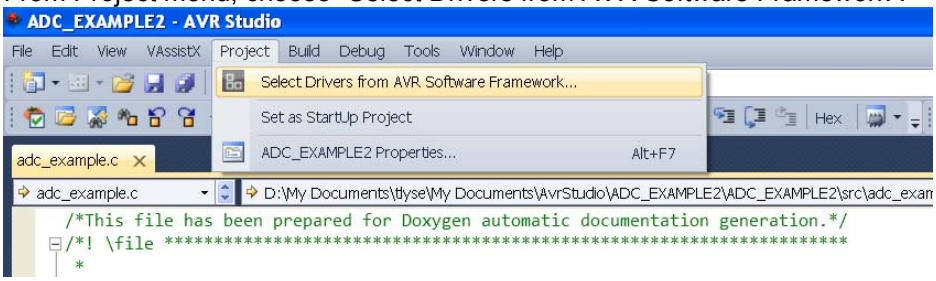
This section describes how to add a USB composite device in an existing project:

1. Import USB modules.
2. Configure personal USB parameters.
3. Call USB routines to run the USB application.

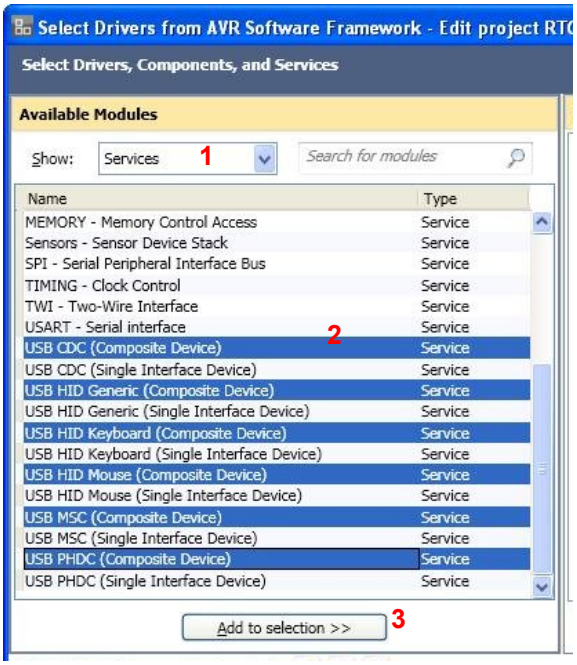
5.1 Import USB modules

To import the USB modules, follow the instructions below:

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



3. Select Services, choose the “USB interface (Composite Device)” required by the application and click on the “Add to selection” button.



5.2 USB configuration

All USB stack configurations are stored in the `conf_usb.h` file in the application module.

The `conf_usb.h` file provided by Atmel AVR Studio 5 contains all default configurations for device and each interface described in next sections. Thus, the user can easily modify these parameters.

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).

5.2.1 USB high level configuration

The high level configurations are simple and do not require any specific USB knowledge. These are already described in following application notes for each USB modules: UDC, UDI and UDD.

The UDC configuration possibilities are described in the Atmel AVR4900 application note in the section 7.1.1: USB device configuration".

The UDD configuration possibilities are described in the Atmel AVR4900 application note in the section 7.1.3: USB drivers configuration".

The UDI configuration possibilities are described in the Atmel AVR4903 to AVR4909 application notes in the section "USB configuration".

5.2.2 USB low level configuration

The low level configuration is required for a composite device.

Three general parameters must be configured for the USB device:

- USB_DEVICE_EP_CTRL_SIZE: endpoint control size. This must be:
  - 8 for low speed
  - 8, 16, 32 or 64 for full speed device (8 is recommended to save RAM)
  - 64 for a high speed device
- USB_DEVICE_NB_INTERFACE: Total Number of interfaces on this USB device
- USB_DEVICE_MAX_EP: Total number of endpoints on this USB device. This must include each endpoint for each interface

Example for a USB composite device mouse and CDC:

```
      //! Device configuration
#define  USB_DEVICE_EP_CTRL_SIZE       64 // Control endpoint size
#define  USB_DEVICE_NB_INTERFACE       3 // 1 for mouse + 2 for CDC
#define  USB_DEVICE_MAX_EP             4 // 1 for mouse + 3 for CDC
```

Two types of parameters are required for each USB interface:

- UDI_X_IFACE_NUMBER: The interface index of an interface starting from 0
- UDI_X_EP: The endpoint number starting from 1. This must be defined together with the USB direction USB_EP_DIR_IN or USB_EP_DIR_OUT

The above values must be defined following a simple number enumeration.

NOTE

_X_: stands for the interface name: HID_MOUSE, CDC_COMM, CDC_DATA, MSC, HID_KEYBOARD, PHDC, …

Example for a USB composite device mouse and CDC:

```
      // Interface number definitions
#define  UDI_HID_MOUSE_IFACE_NUMBER    0 // Mouse interface number
#define  UDI_CDC_COMM_IFACE_NUMBER     1 // CDC COMM interface number
#define  UDI_CDC_DATA_IFACE_NUMBER     2 // CDC DATA interface number

      // Endpoint number definitions
#define  UDI_HID_MOUSE_EP_IN      (1 | USB_EP_DIR_IN)    // Mouse
#define  UDI_CDC_DATA_EP_IN       (2 | USB_EP_DIR_IN)    // CDC TX
#define  UDI_CDC_DATA_EP_OUT      (3 | USB_EP_DIR_OUT)   // CDC RX
#define  UDI_CDC_COMM_EP          (4 | USB_EP_DIR_IN)    // CDC Notify
```

5.2.3 USB descriptors

Each USB interface class in ASF provides one or more descriptors*. These are used to fill the following lists of descriptors dedicated to an application:

```
//! List of descriptor structures
UDI_COMPOSITE_DESC_T
//! List of descriptors for Full Speed or Low speed
UDI_COMPOSITE_DESC_FS
//! List of descriptors for high speed
UDI_COMPOSITE_DESC_HS
//! List of Interface APIs corresponding at interface descriptors
UDI_COMPOSITE_API
```

The descriptors order in these four lists must be the same that order defined by interface index, see Section 5.2.2 on page 7.

NOTE

*The descriptors are defined in the Atmel AVR4903 to AVR4909 application notes in the section “Interface in a USB composite device”.

Extract of `conf_usb.h` for a USB composite device mouse and CDC:

```
//! Define structure of composite interfaces descriptor
#define UDI_COMPOSITE_DESC_T            \
    udi_hid_mouse_desc_t udi_hid_mouse  \ // interface index 0
    usb_iad_desc_t udi_cdc_iad;         \ // interface association
    udi_cdc_comm_desc_t udi_cdc_comm;   \ // interface index 1
    udi_cdc_data_desc_t udi_cdc_data;     // interface index 2


//! Fill composite interface descriptors for Full Speed
#define   UDI_COMPOSITE_DESC_FS                  \
    .udi_hid_mouse            = UDI_HID_MOUSE_DESC, \
    .udi_cdc_iad              = UDI_CDC_IAD_DESC, \
    .udi_cdc_comm             = UDI_CDC_COMM_DESC, \
    .udi_cdc_data             = UDI_CDC_DATA_DESC


//! Fill composite interface descriptors for High Speed
#define   UDI_COMPOSITE_DESC_HS                  \
    .udi_hid_mouse            = UDI_HID_MOUSE_DESC, \
    .udi_cdc_iad              = UDI_CDC_IAD_DESC, \
    .udi_cdc_comm             = UDI_CDC_COMM_DESC, \
    .udi_cdc_data             = UDI_CDC_DATA_DESC


//! Fill Interface APIs corresponding at interfaces descriptors
```

```
#define    UDI_COMPOSITE_API  \
    &udi_api_hid_mouse, \
    &udi_api_cdc_comm, \
    &udi_api_cdc_data
```

5.3 USB implementation

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

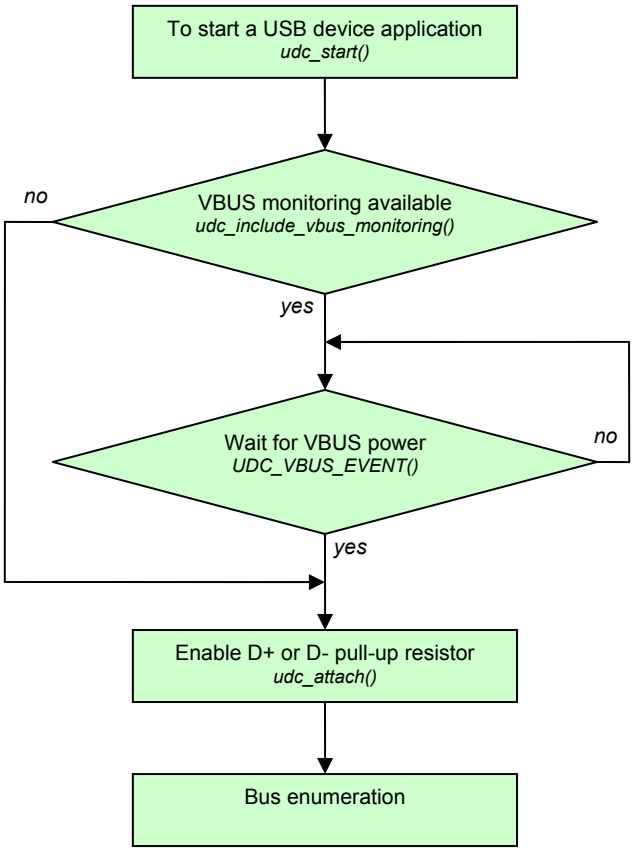
The implementation is comprised of three steps:

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

5.3.1 USB device control

Only two function calls are needed to start a USB device application, see Figure 5-1:

Figure 5-1. 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();
   }
}
```

5.3.2 USB interface control

After the device enumeration (Detecting and identifying USB devices), the USB Host starts the device configuration. When a USB interface is accepted by the USB Host, the corresponding callback function UDI_X_ENABLE_EXT() is called.

When the USB device is unplugged or is reset by the USB Host, the USB interfaces are disabled and the UDI_X_DISABLE_EXT() callback functions are called.

Recommendation about content of these callback functions is available in the application note corresponding to the related USB device class: Atmel AVR4903 up to AVR4009 section “USB interface control”.

NOTE

_X_ : stands for the interface name: HID_MOUSE, CDC_COMM, CDC_DATA, MSC, HID_KEYBOARD, PHDC, …

USB Composite Device example using HID and CDC interfaces:

```
<conf_usb.h>
#define UDI_HIDMOUSE_ENABLE_EXT() \
       mouse_enable()
#define UDI_CDC_ENABLE_EXT() \
       cdc_enable()
...

<main C file>:
cdc_enable() {
   // Open UART and enable UART transition interrupts
   …
   return true;
}
mouse_enable() {
   // Enable sensor and start process to scan sensor
   …
}
...
```

5.3.3 USB data transfer

Finally, developer must use the specific interface functions which come with a dedicated USB class added in the project.

Refer to the Atmel AVR4903 to AVR4909 application notes, section “USB control” to know the specific interface function available.

Example:

```
scheduler() {
   if (is_a_board_button_press()) {
       // Send a button event on USB Line via HID Mouse
       udi_hid_mouse_btnleft(HID_MOUSE_BTN_DOWN);
       // Send a data on USB line via CDC
       udi_cdc_putc(0x55);
   }
   ...
}
```

Basis of the project is now completed. All interface functions are accessible from the user level to handle any USB composite device application.

6 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

8445A-AVR-10/11