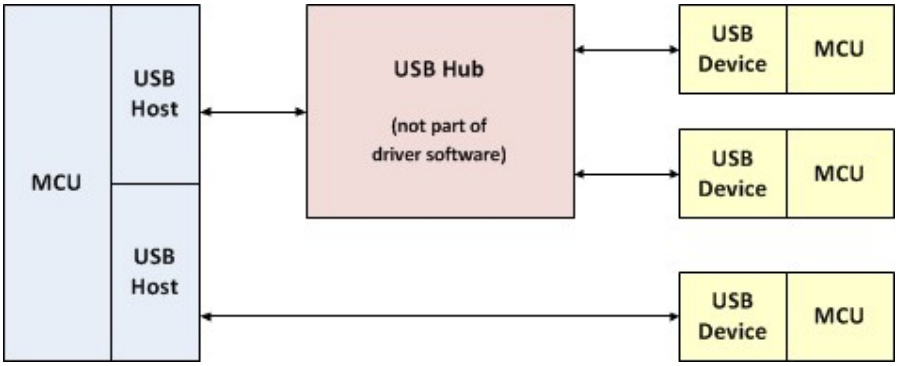
USB

The **Universal Serial Bus** (USB) implements a serial bus for data exchange. It is a host controlled, plug-and-play interface between a USB host and USB devices using a tiered star topology. In microcontroller (MCU) applications, the interface is often used to connect a device to a host for data exchange or control purposes.

Block Diagram

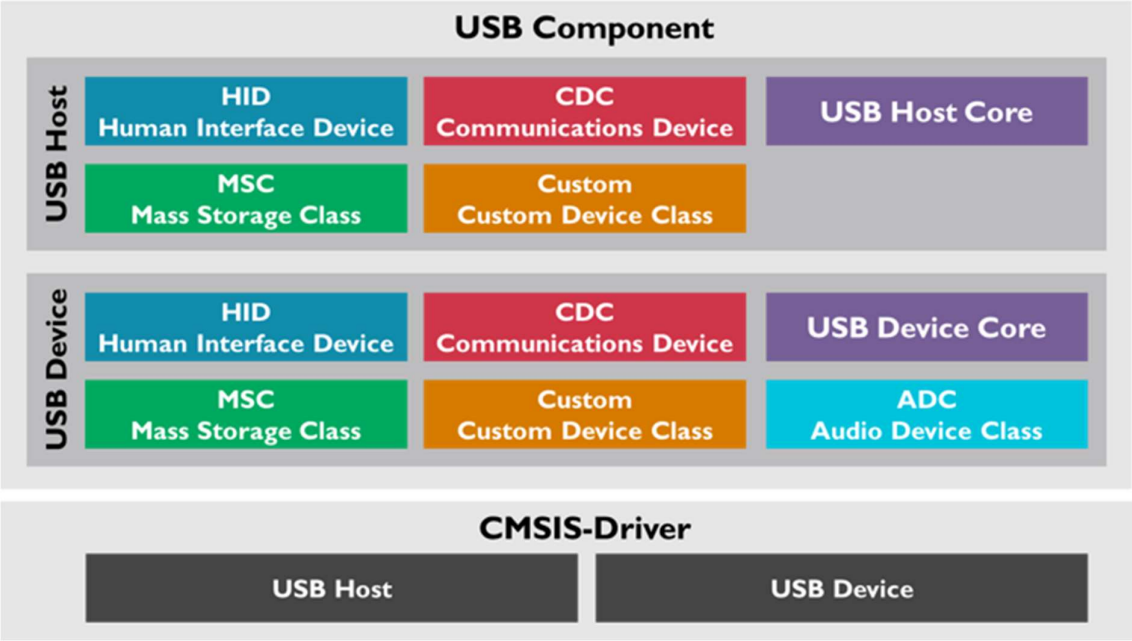
Typically only one USB Device is connected to a USB Host. If several USB devices must be connected to the same USB host, then the connection must be done via a USB hub.



Simplified USB Schema

USB Component Documentation

The **USB Component** enables you to create **USB Device** and **USB Host** applications and is part of CMSIS-Middleware. The USB Protocol is handled by the USB Component, so that developers can focus on their application needs.



Structure of the USB Component

The **USB Component** is structured as follows:

- USB Host is used to communicate to other USB Device peripherals over the USB bus.
- USB Device implements a device peripheral that can be connected to an USB Host.
- The following USB Classes are supported:
  - Human Interface Device (HID)
  - Mass Storage Class (MSC)
  - Communication Device Class (CDC)
  - Audio Device Class (ADC) (USB Device only)
  - Custom Class (for implementing new or unsupported USB Classes)
  - **Composite USB Devices** supporting multiple device classes can be implemented.
- USB (API) for USB Host and USB Device provide the interface to the microcontroller peripherals.

Table of Contents

This chapter describes the software structure of the USB Host Component and explains its use for creating a USB Host application. This software component is available to users of MDK-Professional only.

The USB Host Component simplifies software development of microcontroller systems that allow to connect USB Devices. The attributes of the USB Host Component are:

- Complies with the USB 2.0 specification.
- Support for HID, MSC, CDC, and Custom USB Device Classes to be connected to the USB Host.
- Support for control, interrupt and bulk transfer types.

USB Host Structure

Control Transfers

Control Transfers are bi-directional transfers reserved for the host to send and request configuration information to and from the device using the IN and OUT Endpoint 0.

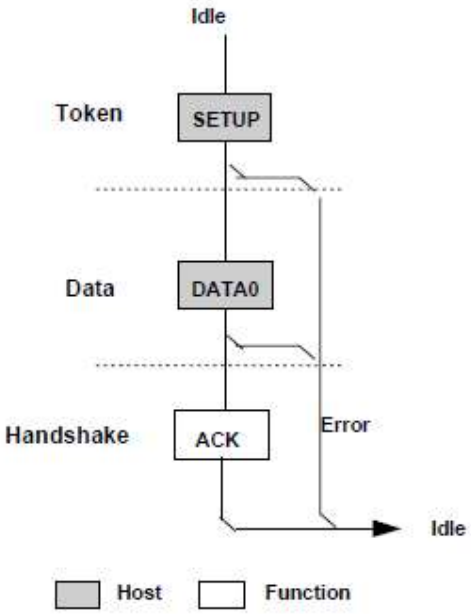
Each Control Transfer consists of 2 to several transactions. The maximum packet size for the control endpoint data is:

- 8 bytes for low-speed
- 8, 16, 32, or 64 bytes for full-speed
- 64 bytes for high-speed

In general, the application software does not use this type of transfer.

Control Transfers have three stages:

1. The **SETUP** stage carries 8 bytes called the Setup packet, defining the request, and specifying how many data should be transferred in the DATA stage.



Control SETUP Transaction Format

2. The **DATA** stage is optional. If present, it always starts with a transaction containing a DATA1 packet. Then, the transaction type alternates between DATA0 and DATA1 until all required data have been transferred.
3. The **STATUS** stage is a transaction containing a zero-length DATA1 packet. If the DATA stage was IN, then the STATUS stage is OUT, and vice-versa.

Interrupt Transfers

Interrupt Transfers have a limited latency to or from a device. In USB, an Interrupt Transfer, or Interrupt Pipe, has a defined polling rate between:

- 1ms and 255ms for full and low-speed
- 125µs to 4096ms for high-speed endpoints.

The maximum packet size for the interrupt endpoint data is:

- 64 or less bytes for full-speed
- 1024 or less bytes for high-speed

The developer can define how often the host can request a data transfer from the device.

For example, for a mouse, a data transfer rate at every 10 ms can be guaranteed. However, defining the polling rate does not guarantee that data will be transferred every 10 ms, but rather that the transaction will occur somewhere within the tenth frame. For this reason, a certain amount of timing jitter is inherent in an USB transaction.

Typically, Interrupt Transfer data consists of event notifications, characters, or coordinates from a pointing device.

Bulk Transfers

Bulk Transfers are used for data which are not of the type Control, Interrupt, or Isochronous. Reliable exchange of data is ensured at the hardware level using error detection.

Data are transferred in the same manner as in Interrupt Transfers, but have no defined polling rate. Bulk Transfers take up all the bandwidth that is available after the other transfers have finished. If the bus is very busy, then a Bulk Transfer may be delayed.

The maximum packet size for the bulk endpoint data is:

- 8, 16, 32 or 64 bytes for full-speed
- 512 bytes for high-speed

For low-speed and full-speed endpoints the following is valid: If the bus is idle, multiple Bulk Transfers can take place in a single 1ms frame (Interrupt and Isochronous Transfers are limited to a maximum of one packet per frame).

For example, Bulk Transfers send data to a printer. As long as the data is printed in a reasonable time frame, the exact transfer rate is not important.

USB Device

Table of Contents

This chapter describes the software structure of the USB Device Component and its use for creating applications. The USB Device Component simplifies the software development of microcontroller systems that interface to a USB Host.

Attributes of the USB Device Component:

- Supports Low-Speed, Full-Speed and High-Speed.
- Supports standard USB Classes with multiple device class instances.

- Supports composite devices. Combine USB Device Classes to create a composite device.
- Supports multiple USB Devices on a single microcontroller with more than one USB Device controller.
- Provides User code template for implementing the USB Device functionality.
- Provides an user-friendly configuration file for each device class to generate USB Descriptors.
- Flexibly assigns USB Endpoints to the microcontroller USB Device peripheral.
- Provides USB Device Examples to show the use of the software stack.

For interfacing to an USB Host Computer, additional software may be required. Page Interface to an USB Host Computer shows an example for such a software running on Windows PCs.

USB Device peripherals can have one or more of the following USB Device Classes:

- Audio Device Class (ADC) is used to exchange streaming audio data between the USB Host and the USB Device.
- Communication Device Class (CDC) provides virtual communication port functionality to the USB Host.
- Human Interface Device (HID) is typically used to implement a keyboard, joystick, or mouse. The HID Class can be also used for low bandwidth data exchange.
- Mass Storage Class (MSC) is used to connect various storage devices to an USB Host. Mass Storage Class media can be an SD card, internal or external Flash memory, or RAM.
- Custom Class is used to implement either a standard or a vendor specific USB Device Class.

Generic information about USB Device Classes can be found on the USB-IF's Approved Class Specification Documents page.

Multiple RTE Component instances can interface with more than one USB Controller or can implement multiple USB Device Classes. RTE Component instances are numbered. The number is appended to the RTE Component name, related configuration files, and user code templates. Each RTE Component has a separate configuration file. For example, for HID 0 and HID 1 the configuration files have the name **USB_Config_HID_0.h** and **USB_Config_HID_1.h**.

USB Transfer Rates

USB uses two wires to supply power and two wires to transfer signals. The following data transfer rates are supported:

Performance	Attributes	Applications
Low-Speed: 1.5 Mbits/s	Lower cost Hot-pluggable Multiple peripherals	interactive devices: mouse, keyboards, game peripherals

Full-Speed: 12 Mbits/s	Low cost Hot-pluggable Multiple peripherals Guaranteed latency Guaranteed bandwidth	phone, audio, compressed video, printers, scanners
High-Speed: 480 Mbits/s	Guaranteed latency High bandwidth	video, mass storage

Note

- SuperSpeed - 5 Gbits/s - is not covered in this document.

HID: Human Interface Device Class

The **Human Interface Device Class (HID)** is mainly used for devices that allow human control over a PC. Using these devices, the host is able to react on human input (e.g. movements of a mouse or keypresses). This response has to happen quickly, so that the computer user does not notice a significant delay between his action and the expected feedback. Typical examples for HID class devices are:

- Keyboards and pointing devices (such as mouse devices, joysticks and trackballs)
- Front-panel controls (for example buttons, knobs, sliders, and switches)
- Simulation devices (such as steering wheels, pedals, other VR input devices)
- Remote controls and telephone keypads
- Other low data-rate devices that provide for example environmental data (such as thermometers, energy meters or even bar-code readers)

A detailed description about HID is provided by the USB Implementers Forum (USB-IF).

MSC: Mass Storage Class

The **Mass Storage Class (MSC)** is mainly used for devices that allow access to their internal data storage. Typical examples for MSC class devices are:

- External hard drives (HDD)
- External optical drives (such as CD or DVD drives)
- Portable Flash memory devices
- Solid-state drives (SSD)
- Digital cameras
- Card readers
- Mobile phones

A detailed description about MSC is provided by the USB Implementers Forum (USB-IF).

CDC: Communication Device Class

The **Communication Device Class (CDC)** supports a wide range of devices that can perform telecommunications and networking functions. Examples for communications equipment are:

- *Telecommunications* devices, such as analog phones and modems, ISDN terminal adapters, digital phones, as well as COM-port devices
- *Networking* devices, such as ADSL and cable modems, as well as Ethernet adapters and hubs

A detailed description about CDC is provided by the USB Implementers Forum (USB-IF).

ADC: Audio Device Class

USB peripherals that support the Audio Device Class send or receive audio, voice, and other sound-related functionality (such as control data for the audio equipment). These audio functions often come with other functions such as video or storage. Together, they make up a composite device (e.g. a DVD-ROM player that can provide video, audio, data storage and transport control functionality). Audio class devices use isochronous data transfer for audio streams. Digitally encoded music data can be delivered using bulk transfers.

A detailed description about ADC is provided by the USB Implementers Forum (USB-IF).