MCUXSDKUSBSHOSTCOMPUG

MCUXpresso SDK USB Stack Composite Host User's Guide

Rev. 3 — 11 July 2022 User guide

Document information

Information	Content
Keywords	MCUXSDKUSBSHOSTCOMPUG, USB Stack, Composite Host, USB
Abstract	This document describes steps to implement a host that supports multiple devices based on the MCUXpresso SDK USB stack.

1 Overview

This document describes steps to implement a host that supports multiple devices based on the MCUXpresso SDK USB stack.

The USB Stack provides one host demo that supports HID mouse + HID keyboard. A user may need a host to meet its requirements, such as the ability to support different class devices like supporting an HID and an MSD device simultaneously. This document provides a step-by-step guide to create a customizable host that supports multiple devices.

2 Introduction

Unlike the composite device that requires many steps, implementing a host that supports multiple devices is simple. The event callback function of host and class can handle attach, enumeration, and detach processing for all the devices. The process flow for this is shown in Figure 1. This figure shows a host supporting two classes, which is the same as a host supporting one class. All class-specific functionality for the devices is achieved in the class-specific task polling in the main function. The user only needs to focus on the modification of these two points.

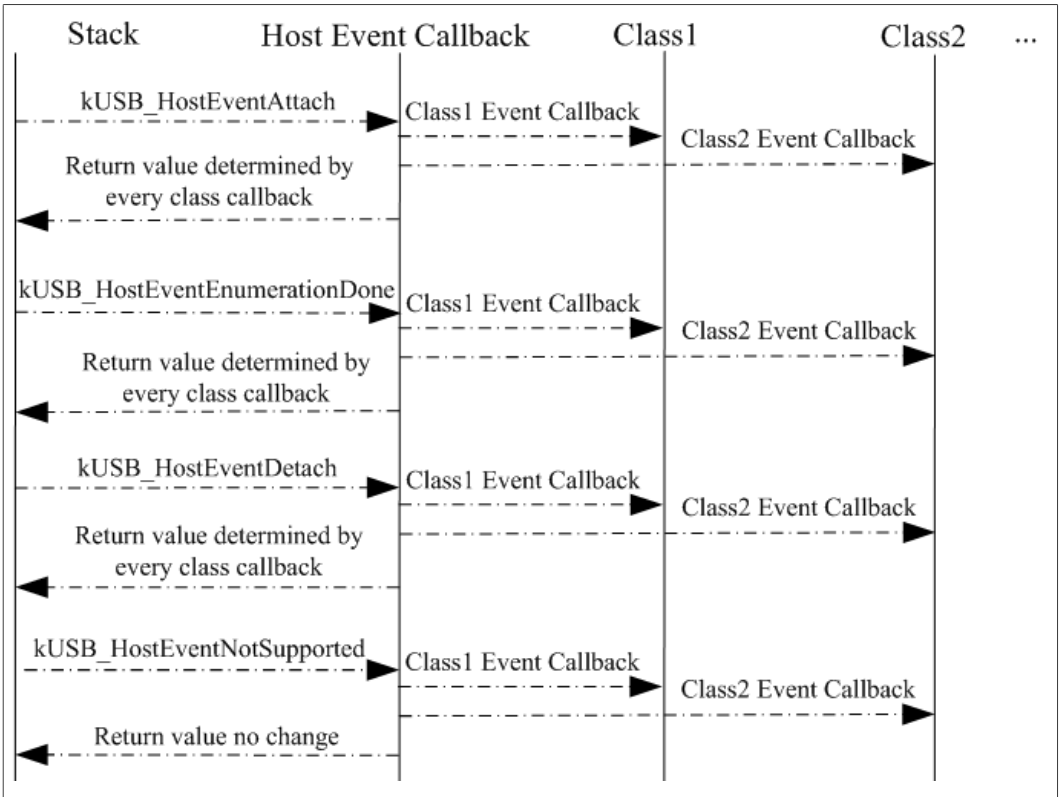

Figure 1. Process flow of event callback

3 Detailed steps

Before developing the host that supports multiple devices, the user needs to determine:

1. How many classes this host needs to support.

2. How many subclasses for every class. For example, the HID class may include HID mouse and HID keyboard.

The code change for the host that supports HID mouse and HID keyboard is similar to that of the host supporting CDC virtual com and HID mouse.

3.1 Host event handle function

The USB_HostEvent is a common handle function for attach, unsupported device, enumeration, and detach event. This function needs to call the class-specific event handle function. When the host only supports CDC devices, the USB_HostEvent function is the following:

```
usb_status_t USB_HostEvent(usb_device_handle deviceHandle,
usb_host_configuration_handle configurationHandle,
uint32_t event_code)
{
usb_status_t status;
status = kStatus_USB_Success;
switch (event_code)
{
case kUSB_HostEventAttach:
status = USB_HostCdcEvent(deviceHandle, configurationHandle, event_code);
/* here add the new device’s event handle function */
break;
case kUSB_HostEventNotSupported:
usb_echo("device not supported.\r\n");
break;
case kUSB_HostEventEnumerationDone:
status = USB_HostCdcEvent(deviceHandle, configurationHandle, event_code);
/* here add the new device’s event handle function */
break;
case kUSB_HostEventDetach:
status = USB_HostCdcEvent(deviceHandle, configurationHandle, event_code);
/* here add the new device’s event handle function */
break;
default:
break;
}
return status;
}
```

To support other devices, add the corresponding class-specific event handle function. Additionally, it is necessary to add the local variable to receive the return value of every event handle function. The return value of USB_HostEvent should be changed according to the following occasions:

1. kUSB_HostEventAttach: if the return values for all of the class-specific event handle functions are kUSB_HostEventNotSupported, the return value of USB_HostEvent is kUSB_HostEventNotSupported.
2. kUSB_HostEventNotSupported: no change.
3. kUSB_HostEventEnumerationDone: if the return values for all of the class-specific event handle functions are not kStatus_USB_Success, the return value of USB_HostEvent is kStatus_USB_Error.
4. kUSB_HostEventDetach: if the return values for all of the class-specific event handle functions are not kStatus_USB_Success, the return value of USB_HostEvent is kStatus_USB_Error.

3.2 Class-specific device task

The main function needs to schedule every supported device's task. If the host only supports CDC devices, the class-specific task in the main function is as follows:

```
int main(void)
{
BOARD_InitHardware();
APP_init();
while (1)
{
USB_HostTaskFn(g_hostHandle);
/* cdc class task */
USB_HosCdcTask(&g_cdc);
/* here add the new device's task */
}
}
```

4 Host MSD command + CDC virtual com example

This section provides a step-by-step example for how to implement a host that supports CDC virtual com and MSD command. This example is based on the existing host CDC virtual com example.

4.1 USB component files

Add the usb_host_msd component files, the usb_host_msd_ufi source file, and the host_msd_command component files into the current project. Normally, the host_msd_command component should be in the source folder, shown in Figure 2. The usb_host_msd component and the usb_host_msd_ufi source file should be located in the class folder showing in the Figure 3.

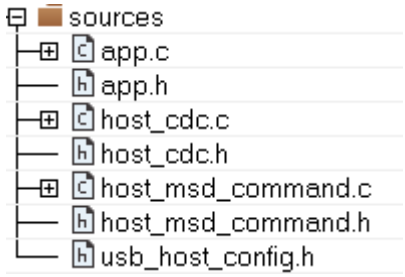

Figure 2. Source folder

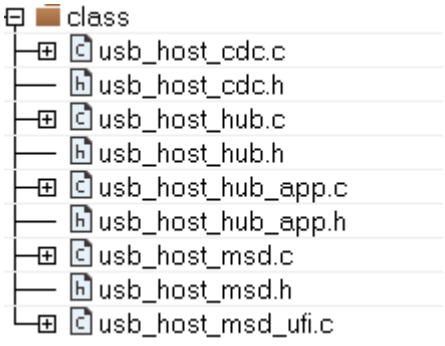

Figure 3. Class folder

4.2 USB_HostEvent function

Add the USB_HostMsdEvent function into the USB_HostEvent function.

```
usb_status_t USB_HostEvent(usb_device_handle deviceHandle,
                           usb_host_configuration_handle configurationHandle,
                           uint32_t event_code)
{
    usb_status_t status1;
    usb_status_t status2;
    usb_status_t status = kStatus_USB_Success;
    switch (event_code)
    {
        case kUSB_HostEventAttach:
            status1 = USB_HostCdcEvent(deviceHandle, configurationHandle,
 event_code);
            status2 = USB_HostMsdEvent(deviceHandle, configurationHandle,
 event_code);
            if ((status1 == kStatus_USB_NotSupported) && (status2 ==
kStatus_USB_NotSupported))
            {
                status = kStatus_USB_NotSupported;
            }
            break;
        case kUSB_HostEventNotSupported:
            usb_echo("device not supported.\r\n");
            break;
        case kUSB_HostEventEnumerationDone:
            status1 = USB_HostCdcEvent(deviceHandle, configurationHandle,
 event_code);
            status2 = USB_HostMsdEvent(deviceHandle, configurationHandle,
 event_code);
            if ((status1 != kStatus_USB_Success) && (status2 !=
 kStatus_USB_Success))
            {
                status = kStatus_USB_Error;
            }
            break;
        case kUSB_HostEventDetach:
            status1 = USB_HostCdcEvent(deviceHandle, configurationHandle,
 event_code);
            status2 = USB_HostMsdEvent(deviceHandle, configurationHandle,
 event_code);
            if ((status1 != kStatus_USB_Success) && (status2 !=
 kStatus_USB_Success))
        {
            status = kStatus_USB_Error;
        }
        break;
    default:
        break;
    }
    return status;
}
```

4.3 Main function task

Add the USB_HostMsdTask function into the main function. The modified code should look like this:

```
int main(void)
{
    gpio_pin_config_t pinConfig;
    BOARD_InitPins();
    BOARD_BootClockRUN();
    BOARD_InitDebugConsole();
    /* enable usb host vbus */
    pinConfig.pinDirection = kGPIO_DigitalOutput;
    pinConfig.outputLogic = 1U;
    GPIO_PinInit(PTD, 8U, &pinConfig);
```

```
    APP_init();
    while (1)
    {
        USB_HostTaskFn(g_HostHandle);
        /* cdc class task */
        USB_HosCdcTask(&g_cdc);
        /* msd class task */
        USB_HostMsdTask(&g_MsdCommandInstance);
    }
}
```

5 Revision history

The following table summarizes the changes done to this document since the initial release.

Table 1. Revision history

Revision number	Date	Substantive changes
0	11/2018	Initial release
1	12/2018	2.4.0 vs 2.5.0
2	05/2020	Updated for MCUXpresso SDK v2.8.0
3	11 July 2022	Editorial and layout updates.

6 Legal information

6.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

6.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

For more information, please visit: http://www.nxp.com

Date of release: 11 July 2022
Document identifier: MCUXSDKUSBSHOSTCOMPUG