MCCI Corporation
3520 Krums Corners Road
Ithaca, New York 14850 USA
Phone +1-607-277-1029
Fax +1-607-277-6844
www.mcci.com

TrueTask® USB for TI AM64xx/AM243x Sitara™ Devices

MCCI Document 971001081
Rev A
Date: 2022-06-27

PROPRIETARY NOTICE AND DISCLAIMER

Document Release History

Rev A	2022-06-27	First release

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

This document summarizes how MCCI supports the Texas Instruments AM64xx and AM243x SOCs with TrueTask USB and the MCCI USB DataPump.

1.1 Referenced Documents

Table 1. References

Reference	File	Comments
[DP-TECH]	MCCI document 971001018	DataPump Technical Overview (link)

2 Overview

TrueTask USB and the MCCI USB DataPump are an extremely robust, complete, portable, high-performance programming environment for USB applications from low-speed USB to USB 3.2 applications. Under continuous development for over 25 years, the package offers a comprehensive combination of features.

- Host, device, and combination host/device product support
- Low speed, full speed, high speed, and SuperSpeed (gen1, gen2, and gen2x2) support
- Complete support for all USB transfer modes (control, bulk, interrupt, isochronous) including high-bandwidth modes
- Type C DRP and USB OTG support
- A broad variety of host and device classes
- Middleware to ease integration into customer applications

Working closely with TI engineers, MCCI has ported this environment to the AM64xx and AM243x platforms, allowing TI customers to easily access MCCI's USB technology and support in their design projects.

Because of the breadth of MCCI's offering, we break the discussion into several sections.

- Device stack integration
- Host stack integration
- Dual role architectures
- Details for specific device classes

TrueTask USB and the DataPump are architected as a tree of protocol modules and drivers, much like a TCP/IP network stack, with stable APIs at each interface. This helps to localize the design sensitivity to supported speeds, supported device classes, even to supported operating systems.

2.1 High Speed vs SuperSpeed support

Many projects only require high-speed or full-speed USB support, and have other uses for the SerDes features for USB. The DataPump's modular architecture makes it easy to choose the desired mode of operation. The decision primarily affects the lower levels of the stack. In some cases, omitting SuperSpeed support simplifies the integration; we call that out explicitly.

To save words, when we write "high speed" we mean all the relevant speeds of the USB 2.0 bus. As a device, the AM64xx/AM243x support high speed and full speed operation. As a USB host, the AM64xx/AM243x can communicate with low speed, full speed, and high-speed devices.

2.2 SDK Integration

Unless we indicate otherwise, all the MCCI code is intended to be used with the Texas Instruments SDK.

The core DataPump can be used with any operating system. Integrations are suppled for both the FreeRTOS port supplied by TI, and the non-RTOS environment.

Some components are FreeRTOS-specific; we'll call those out specifically below.

In the diagrams, we show code that is supplied by TI or the customer.

3 Device Stack

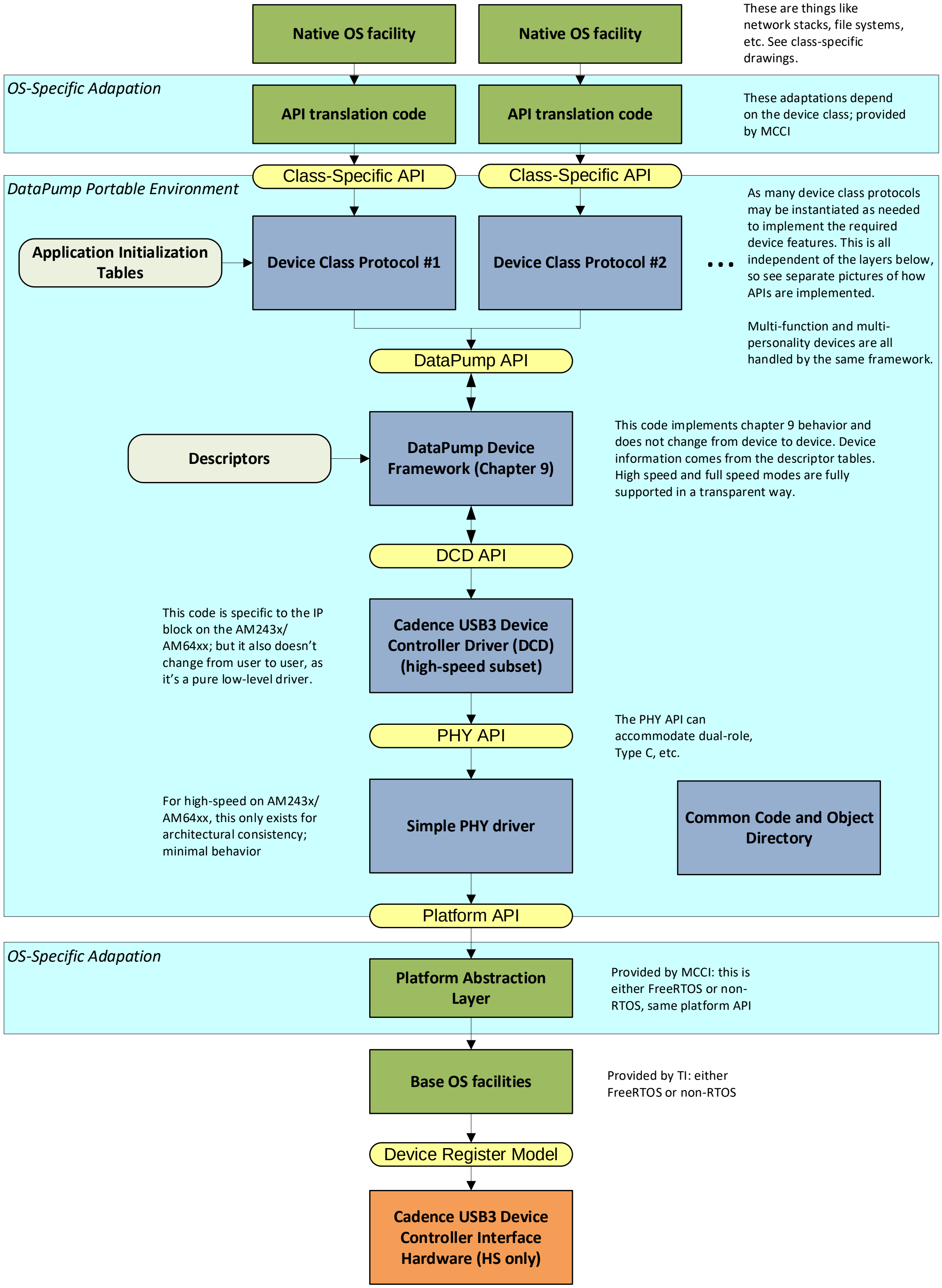
3.1 High Speed Only

The architecture of the high-speed stack is shown in Figure 1. The details of the specific device class implementations are glossed over; we come back to device classes in section 6. Essentially all the code is provided by MCCI, with the exception of the very low level FreeRTOS and non-RTOS primitives, which come from the SDK.

The DataPump device stack is inherently **multifunction** and **multipersonality**. This means that it's easy to make composite USB devices, as well as USB devices that enumerate differently under different circumstances, for example normal operation plus a debug mode, or normal operation plus DFU. Hence, we've shown multiple device class protocol modules at the upper edge. This isn't required; you don't need more than one to make a valid and useful device.

Another important feature that doesn't fit well on diagrams is the ability to vary VID/PID/strings and descriptor sets dynamically at runtime. This is done via an API, registering a function that the DataPump calls in response to host queries.

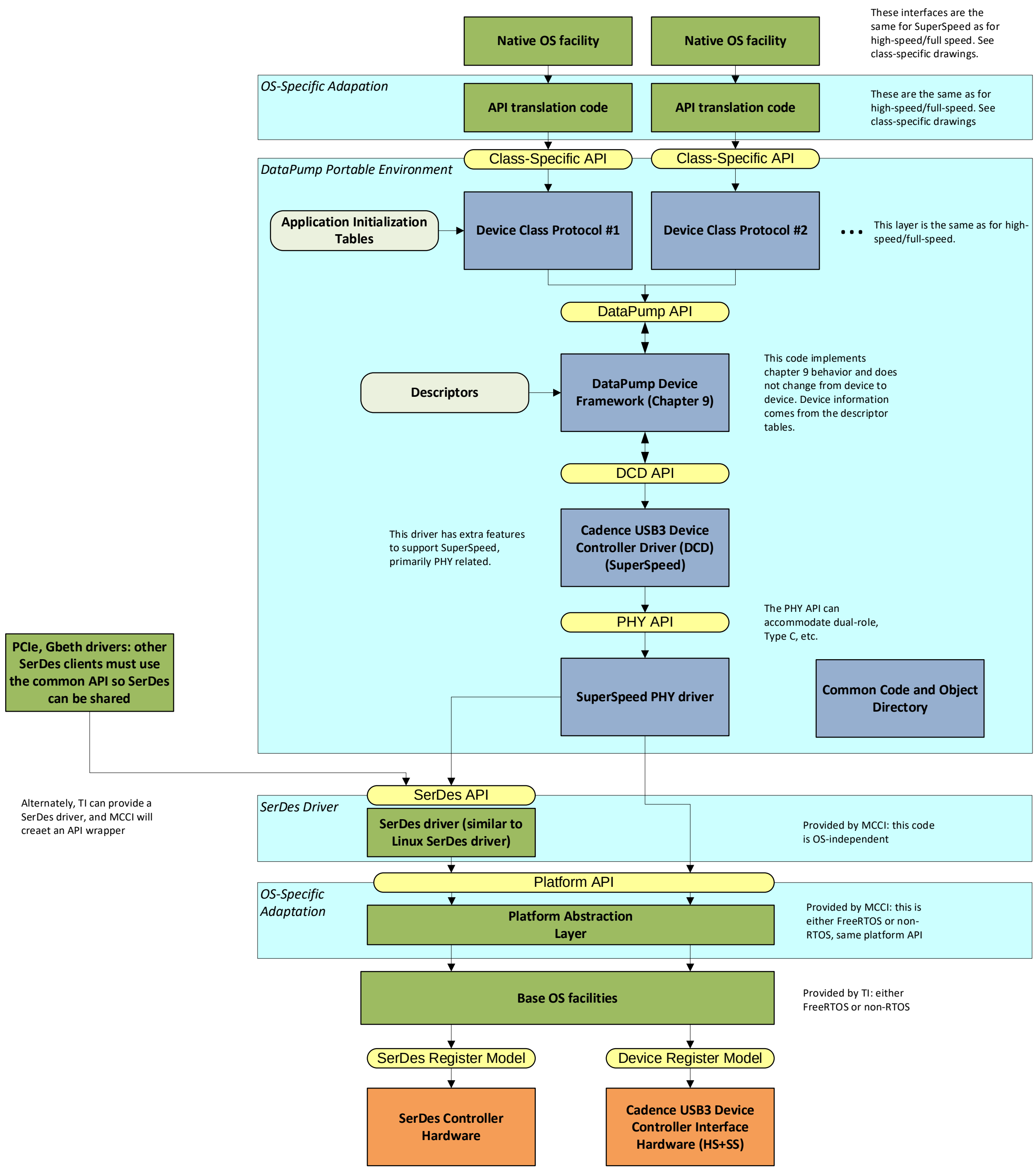
Figure 1. High-Speed Device Architecture



3.2 High Speed plus SuperSpeed

The high-speed plus SuperSpeed configuration is shown in Figure 2. SuperSpeed Device Stack

Figure 2. SuperSpeed Device Stack



The key difference here is that on the TI platform we have to add an interface to set up hardware that's shared between several peripherals (and therefore several different drivers). This hardware is the SerDes; it's used for the SuperSpeed PHY, as well as for PCIe, integrated Gigabit Ethernet, and other functions. MCCI supplies a driver which can be used to initialize the SerDes for USB functions; if you have other drivers using the SerDes, you can use MCCI's driver, too. Alternately, you can use a general SerDes driver from TI or other sources, and MCCI can supply a wrapper to call that driver based on our APIs.

4 Host Stack

4.1 High Speed Only

The architecture of the high-speed variant of the host stack is shown in Figure 3.

It's very similar to the high-speed device stack, in that the SerDes component is not needed. Low speed, full speed, and high-speed devices are supported. A composite driver is supplied, along with a full hub driver and transaction translator support.

4.2 High-Speed plus SuperSpeed

Figure 3. High-Speed Host Stack

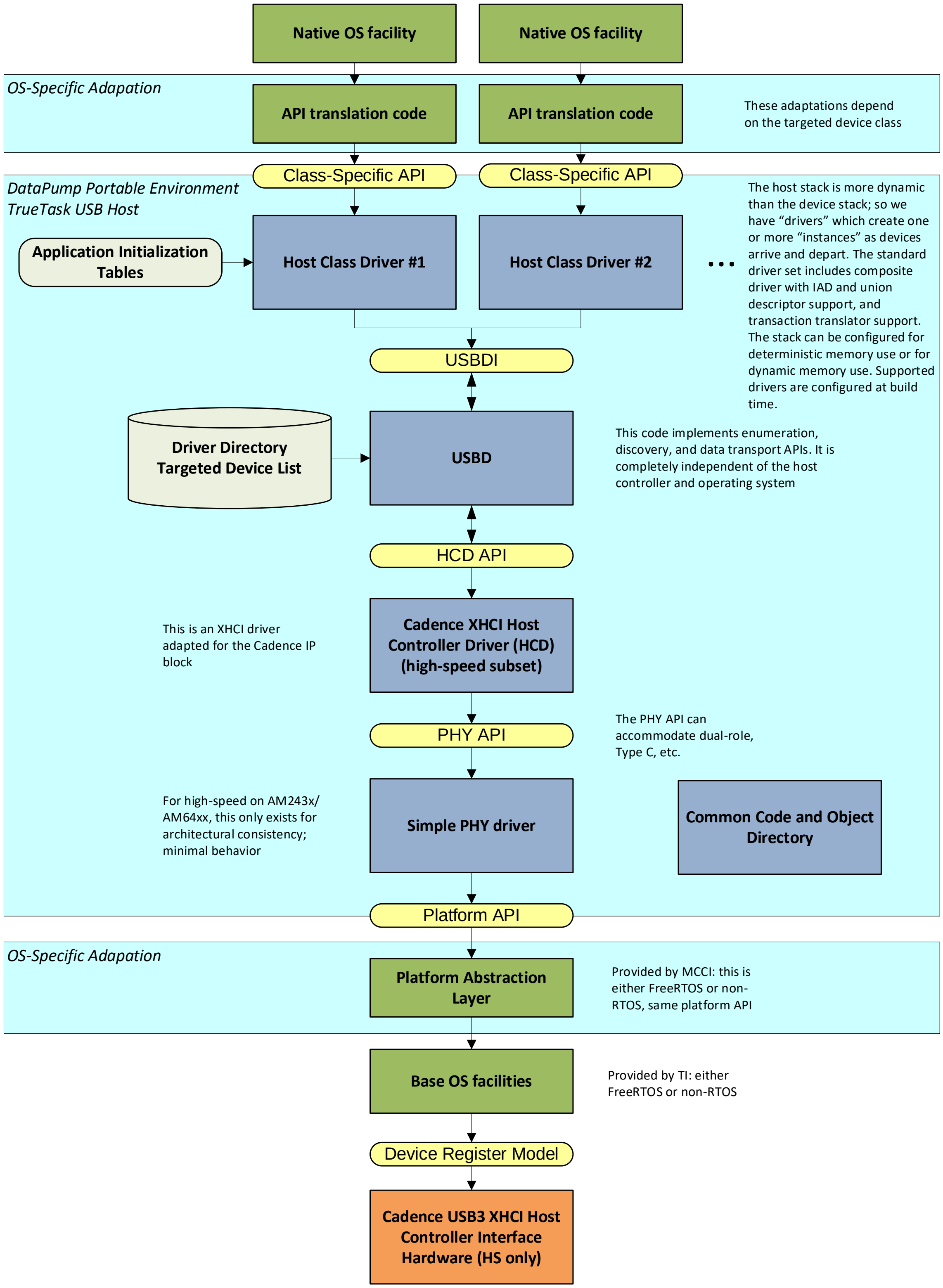
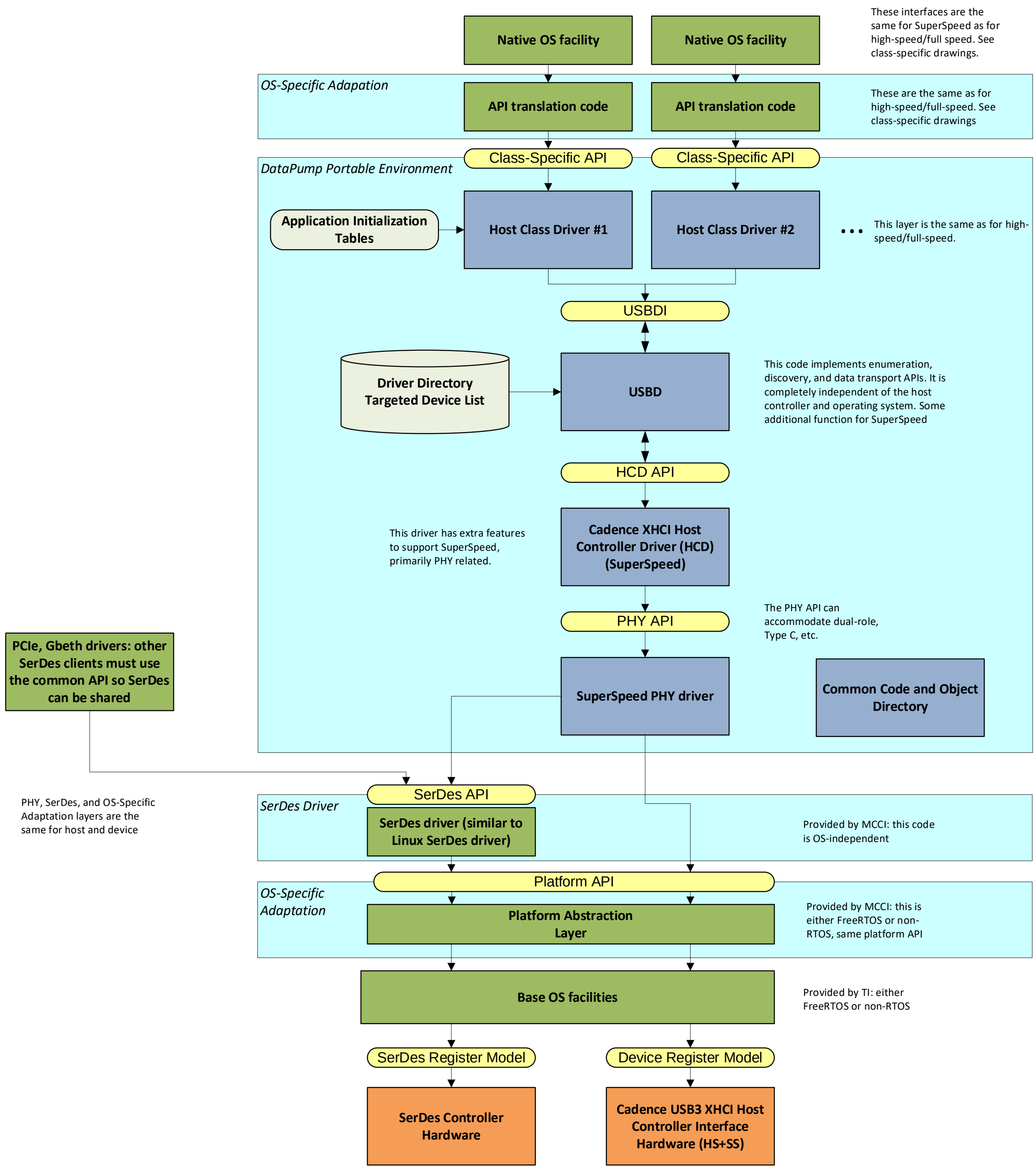


Figure 4. SuperSpeed host stack



5 Dual Role Architectures

Rather than repeat the details of the dual-role architecture here, we refer you to the figures and descriptions in [DP-TECH].

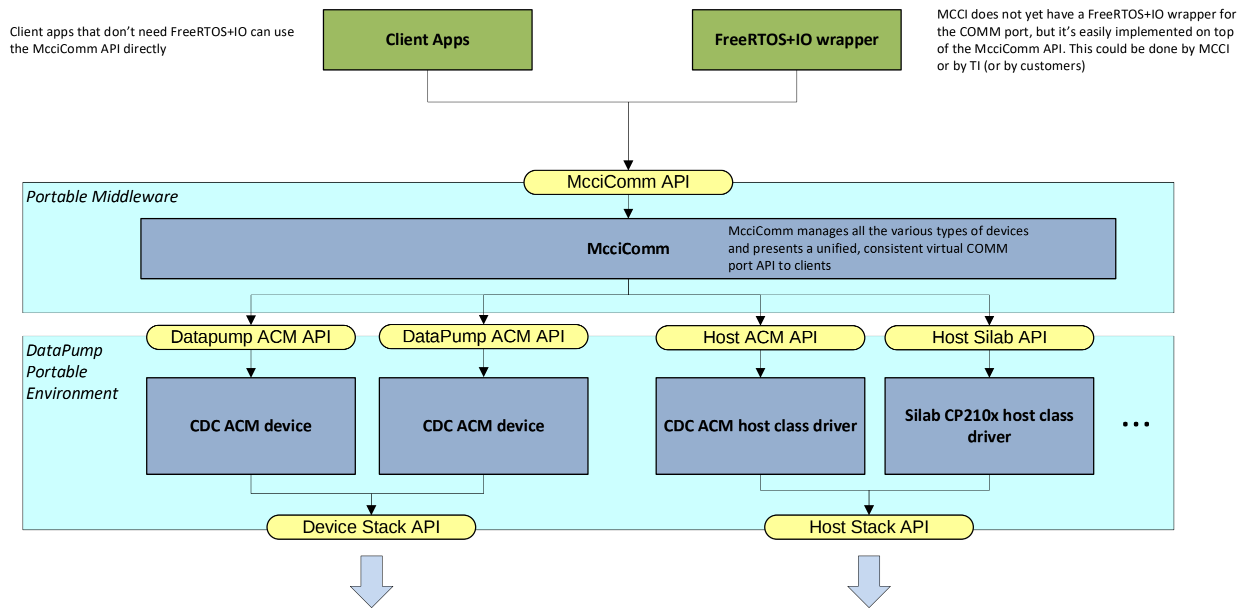
6 Specific Class Support

6.1 Serial (ACM, CP210x)

Serial ports over USB are inherently symmetric: device and host both implement essentially the same abstraction. The MцciComm API defines a single API to support USB host and USB device use cases. As a USB device, it supports devices that expose one or more CDC ACM ports. As a USB host, it is able to discover and operate both ACM-based functions on devices and USB serial ports based on the Silicon Labs CP210x. Because composite driver support includes ACM union descriptors and the Interface Association Descriptor, a wide range of single and multi-function devices can be supported.

The MCCI PortLynq technology[1] is used on the virtual serial ports, translating hot plug events into virtual modem control-line activations, simplifying application implementation and test.

Figure 5. Serial driver (host/device)



6.2 Networking

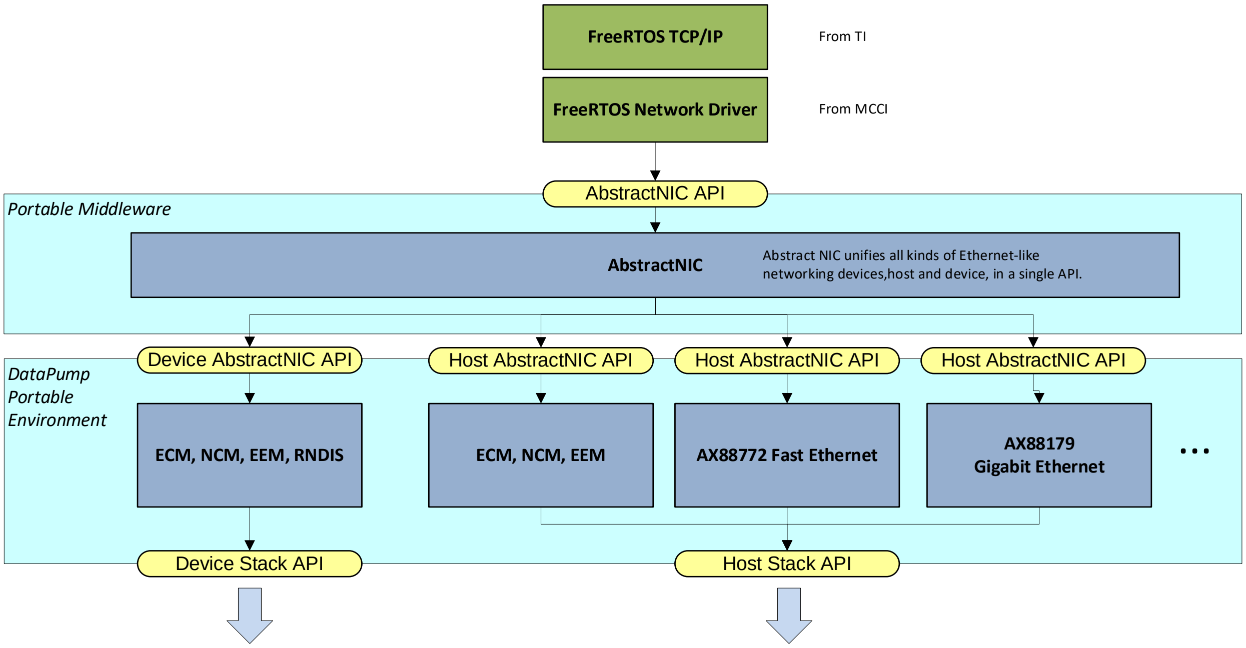
Ethernet networking devices are also symmetric between host and device. MCCI's AbstractNIC API again allows the network interfaces to be unified under a single driver.

As a device, the MCCI stack allows you to choose from a number of network protocols: ECM, EEM, NCM, and RNDIS. As a host, you can configure support for ECM, EEM and NCM devices, as well as for USB to Ethernet adapters using controller chips from ASIX (the AX88772 and the AX88179).

See Figure 6.

[1] US patent 8,346,981

Figure 6. Network class drivers

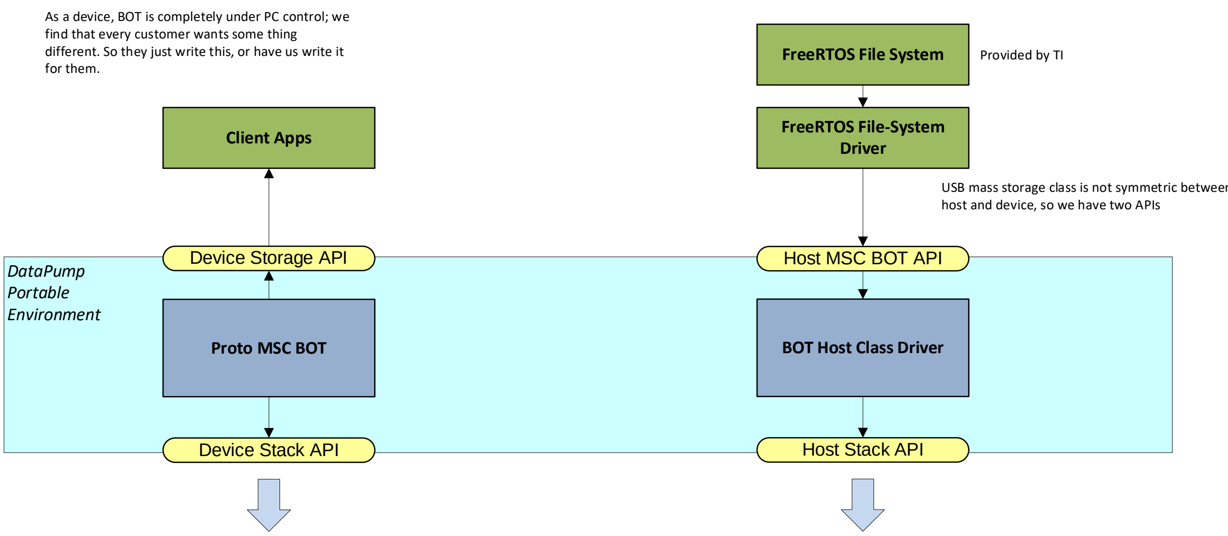


6.3 Mass Storage

Mass storage is not symmetric between host and device. The host stack needs to present devices to the file system, and translate the file system APIs to the block commands understood by the external devices. On the other hand, the device stack needs to receive those block commands, and access some low-level block storage medium. On a device, this is generally a highly custom process.

See

Figure 7. Mass Storage

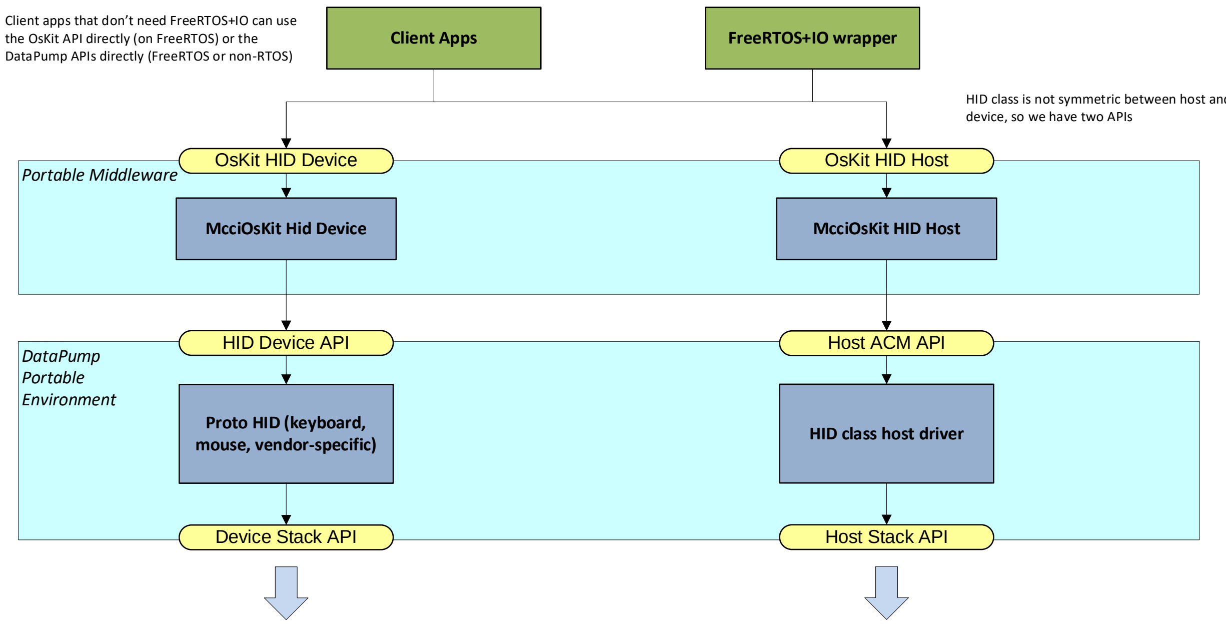


6.4 HID

MCCI supplies HID class drivers for host and device. Again, the APIs are not symmetric. As a device, HID class is generally used for providing a vendor-specific driverless control interface to the host. As a host, HID class is generally used to receive input from things like keyboards and mice.

See Figure 8

Figure 8. HID Class

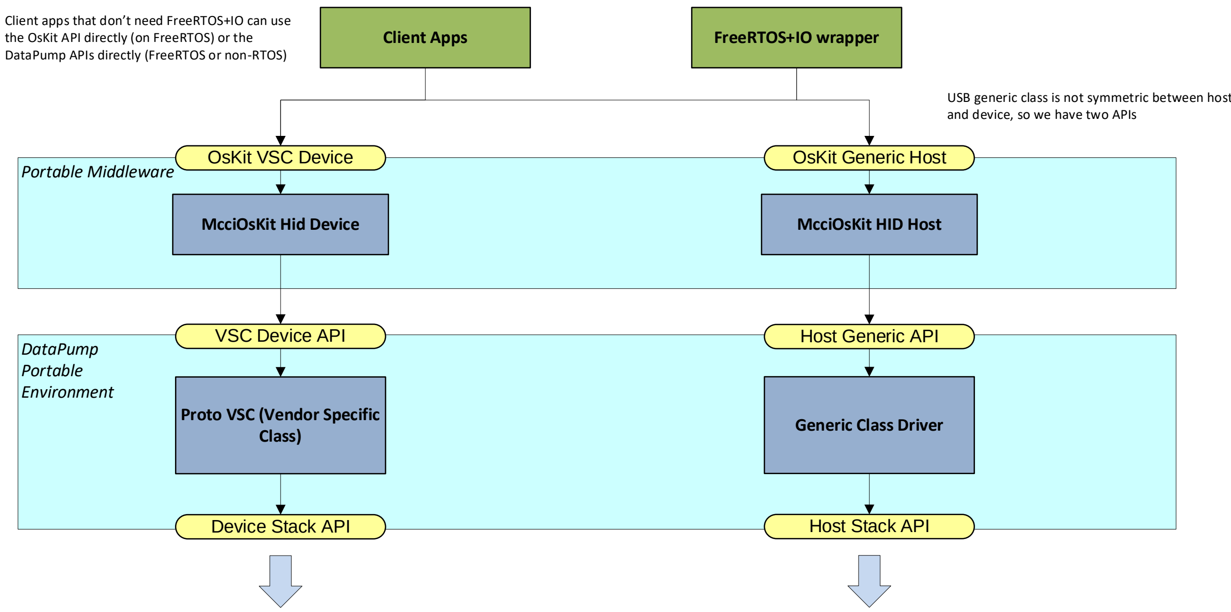


6.5 Generic/Vendor Specific

The Generic/Vendor Specific APIs allow you to create your own custom device class, or to control (as a host) any device in a way that's similar to libusb.

See Figure 9.

Figure 9. Generic/Vendor-Specific



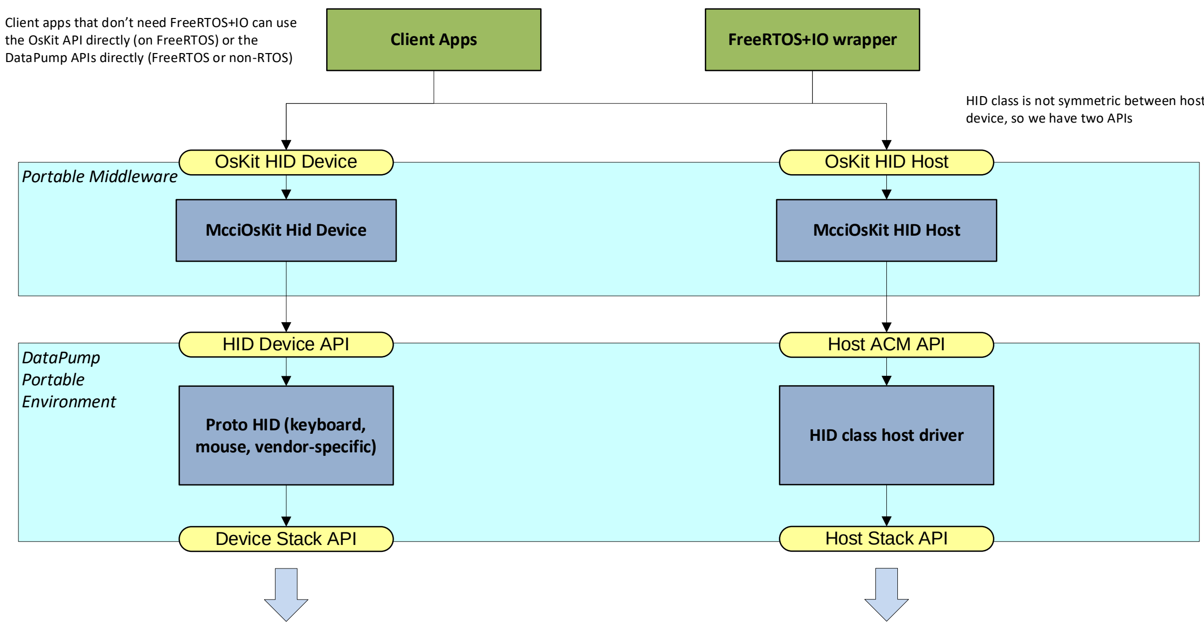
6.6 Audio

MCCI offers complete support for Audio 1.0 and 2.0 on both host and device. Because the control plane is asymmetric, and because data start/stop sequences are different, different APIs are used for host and device.

Isochronous feedback endpoint support is available for high-quality audio applications, along with Codec interface middleware.

See Figure 10.

Figure 10. Audio Class



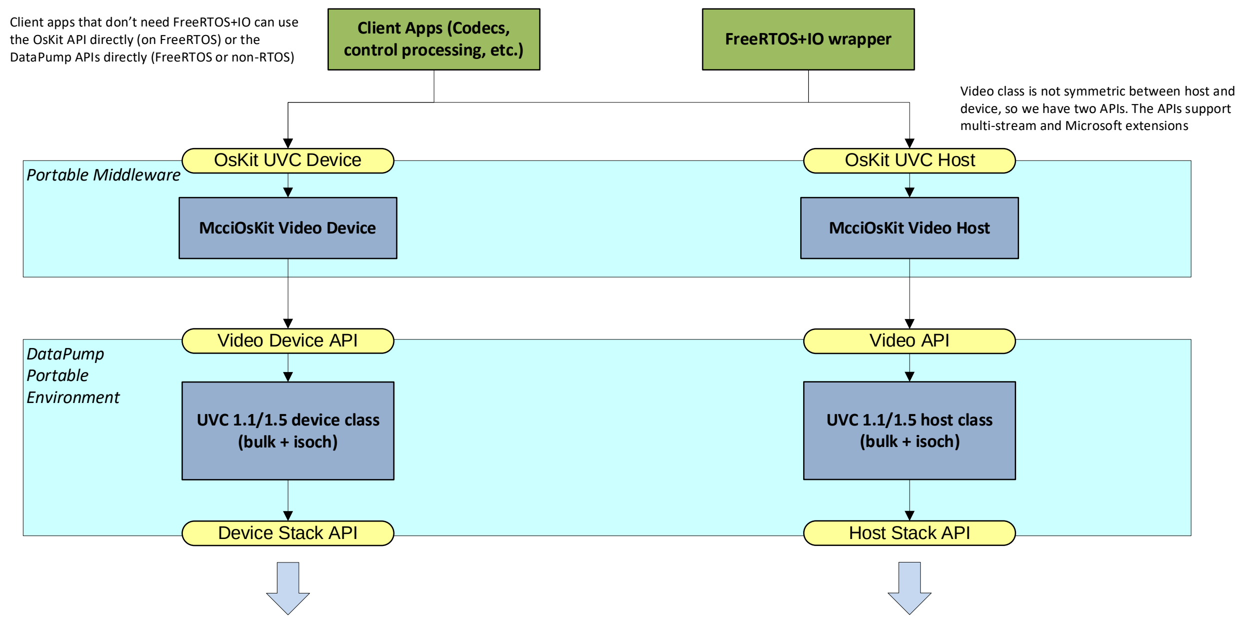
6.7 Video Class (webcam)

MCCI offers complete support for USB Video Class (UVC) 1.0 through 1.5. The stack includes support for multiple streams, metadata in the video stream, bulk and isochronous wrappers, and Microsoft extensions. The customer must supply video CODECs and implement any required behavior for the UVC controls.

The implementation is asymmetric because of the nature of the UVC specification. Devices always

See Figure 11.

Figure 11. Video Class (UVC)



7 Application-Specific Support

7.1 Firmware Update

MCCI offers support for firmware update via USB, using either the device or host stacks.

The implementation is asymmetric because as a device, the Sitara receives data from the USB host following the DFU protocol; as a host, the Sitara must pull data from a USB device.

Based on consultation with the customer, MCCI can provide turn-key solutions, or can assist customers in building a suitable firmware update system.

The system is not limited to these two approaches; for example, it's also possible to use USB comm ports (ACM) or vendor specific devices classes, or TFTP over a USB networking link. MCCI's engineers will work with you to find a solution that meets your product requirements.

Figure 12. Firmware Update Architecture