TrueTask USB / MCCI USB DataPump Technical Overview

MCCI Corporation, July 2015

Introduction

TrueTask USB is MCCI's USB software platform for embedded systems. It's based on the MCCI USB DataPump ("DataPump"), MCCI's portable embedded USB framework. This article gives a technical overview of the DataPump for engineers and technical managers.

The DataPump combines comprehensive USB device support with the industry's most thoroughly verified embedded USB host/USB On-The-Go (OTG) stack. Its modular, reentrant design allows it to be deployed in a number of ways. It may be used as a stand-alone stack; or it may be used to augment, supplement, upgrade, or replace existing USB stacks, while maintaining compatibility with the existing stack.

The modular architecture of the DataPump allows it to scale from the simplest operating environments to the most complex. In the device stack, all memory allocation happens during initialization. The host stack can be configured (for deeply embedded systems) to allocate all memory during initialization, or to allocate and free memory for device instance data while the stack is running.

In addition, by careful abstraction and layering, the components can be used independently. In an OTG environment, the DataPump device component can be used with or without the DataPump host stack.

All MCCI software is designed to serve as a platform for further development. APIs and data structures are stable from version to version, which means that system software built on the DataPump can coded once, then reused as needed.

The careful design and implementation allows the DataPump to be readily deployed without source modification across a wide range of CPU architectures, USB host and device controllers, and operating systems. A single code base supports a range of use cases, from USB 1.1 full-speed to USB 3.1 SuperSpeed Plus products, including OTG, Wireless USB, 802.11ad WSE docks, and Media-Agnostic USB (MA-USB).

The DataPump is the result of over 15 years of continuous development and refinement. It is accompanied by a large suite of test tools, regression tests, and development applications. This ecosystem makes it the strongest engineering solution for supporting multiple USB products from a common source base.

TrueTask USB is a packaging of select components of the DataPump in a form that is easy to use in conjunction with select real-time operating systems. The components are pre-ported and integrated with the RTOS and the target SOC, and delivered as a software component. TrueTask USB inherits the stable APIs of the DataPump, and so insulates customer software from variations between hardware platforms, and variations between TrueTask USB releases.

Product History

Since 1997, over 700 million products have shipped using MCCI USB DataPump technology. Here are some key milestones in the product's history.

1997	First release, targeting 68K and USB 1.0 silicon (device stack only). Support for virtual serial port and loopback protocols. Support for Agere USS-820 silicon.
1998-2001	Primarily used in cable modems. CDC ECM, RNDIS, HID class, mass storage, DFU protocol modules added. Support for Philips D12, Broadcom SOC, StrongArm SOC USB device controllers. ARM, MIPS support.
2002-2004	First cell phone firmware design wins. CDC WMC (ACM, OBEX, Device Management support), MCPC GL-004/005, mass storage protocol modules added. First cellular platform design wins. PowerPC, SPARC support.
2005	High-speed USB device support. PictBridge. Support for Mentor device IP.
2006	Full-speed host/OTG support. Support for Mentor OTG IP. MTP 1.0 device.
2007	High-speed host/OTG support. Support for Synopsys HS OTG IP, ST-E ISP1761, Renesas '597.
2008	Wireless USB device support. Added Audio 1.0 device, USB Video device protocols.
2009	Super speed device and host support. Support for Synopsys USB 3.0 device core, and for xHCI. Support for Link Power Management in device stack. NCM host and device. EEM support for USB-UICC. Support for 64-bit CPUs (AMD x64 and Itanium).
2010	Validation of host stack on Windows; support for HSIC host and device. Support for automatic Link Power Management in embedded and Windows stack. Added UAS device class protocol support.
2011	Support for MBIM device implementations.
2012	"TrueTask USB" embedded USB platform introduced. Kernel-mode USB device and OTG support for Windows. First production LTE modems based on DataPump technology. Synopsys DesignWare USB 3.0 device controller support.
2013	Support for ARC processors, Freescale i.MX6 and Atmel 91SAM9M10 SOCs, PLX USB 3.0 device cores.
2014	Support for SSIC USB; support for FTDI FT900, Renesas RZ1, ST-M STM32F207IG and NXP LPC1850 SOCs. Validated with USB 3.1 xHCI host controllers.
2015	Support for Broadcom 2836 SOC; TrueTask USB selected by Microsoft Corp. for incorporation in Windows 10 IoT Core for Raspberry Pi 2.

MCCI's broad protocol support, standards leadership, and technical excellence are recognized by high-volume consumer product makers around the world, and have made the DataPump the "gold standard" for trouble-free applications of USB across a product line or throughout a corporation.

DataPump Components

The DataPump has the following components.

DataPump Device Stack

The DataPump Device Stack consists of the Device Framework, Device Controller Drivers, support libraries, and device class protocols.

The Device Framework provides functionality common to all USB devices, including standard command support, suspend/resume, link-power-management support, composite device support, multiple device modes, and Microsoft OS Descriptor support.

Device Controller Drivers

Device Controller Drivers (DCDs) provide a common, portable, low-level API to the Device Framework. This API is optimized for high throughput, zero copy DMA operations.

Over thirty DCDs are available. Notable DCDs include:

- Synopsys DesignWare USB 2.0 IP. This DCD supports the industry standard Synopsys high-speed device and OTG IP. Versions 2.6 and later are supported. Depending on the version, PIO, DMA and scatter-gather DMA are supported. Normal USB and HSIC are supported. LPM is supported if supported by the hardware.
- Mentor Inventra. This DCD supports the popular Mentor MUSBMHDRC high-speed OTG core. A variety of PHY architectures allow support for common external PHYs. LPM is supported if supported by the hardware.
- Renesas '597. This DCD supports the Renesas high-speed OTG kit part (for lower volume designs and prototyping) and IP (for SOC applications).
- Synopsys DesignWare USB 3.0. This DCD supports the Synopsys SuperSpeed device IP. Support is included for streams.
- Cadence USBHS-OTG-MPD. USB 2.0 device core with advanced DMA, and multi-device host controller for dual-role and USB On-The-Go applications supporting hubs.
- Cadence USBHS-OTG-SD. USB 2.0 device core with simple DMA. Simplified host controller supports dual-role device and USB On-The-Go applications that don't involve hubs or compound devices.

DataPump Device Class Protocols

Device Class Protocol modules provide the support for device classes. MCCI offers over 20 device class protocols. All protocols can be freely combined to form composite multi-function devices, and to create sophisticated multi-mode devices.

- Audio Class 1.0 and 2.0
- CDC (Communications Device Class) 1.2 Wireless Mobile Communication subclass (WMC) for multi-function 2.5G and 3G handsets

    - CDC WMC Abstract Control Model (ACM), for traditional modems and modem emulation
    - CDC WMC Device Management
    - CDC WMC OBEX (Object Exchange)
    - CDC WMC MDLM (debug ports, vendor specific functions, etc.)
- Abstract NIC family of virtual NICs over USB, with support for:
    - CDC ECM (Ethernet Control Model), for low throughput Ethernet-like networking, targeting cable modems and network bridges
    - CDC EEM (Ethernet Emulation Model), targeting accessories and local peripheral networking
    - CDC NCM (Network Control Model), for high throughput Ethernet-like networking
    - Microsoft RNDIS, for networking applications targeting Microsoft Windows systems
- Device Firmware Update (DFU) 1.1, for firmware update over USB
- Generic class (Vendor Specific Class), for implementing USB device behavior outside the DataPump
- Human Interface Device (HID) 1.1
- Mass Storage Bulk-Only Transport. This protocol is used both for read/write storage applications, and for CD-ROM emulation (for automatic driver installation on Windows and Mac OS X systems).
- MCCI Loopback, for test and performance evaluation
- MCCI Virtual Serial Port (VSP), for migrating RS-232 devices to USB
- Mobile Broadband Interface Module (MBIM) 1.0
- Network Control Model (NCM) 1.0
- Still Image Class
- USB Attached SCSI (UAS) 1.0
- Video Class 1.1

DataPump Device Class Applications

Building on the Still Image Class implementation, MCCI offers PTP, PictBridge and Media Transport Protocol implementations. These implementations include a database suitable for removable media, and the software for automatic media file discovery and indexing.

DataPump Host Stack

The DataPump host stack, like the device stack, consists of the core functionality ("USBD"), host controller drivers (HCDs), and class drivers.

DataPump USBD

The DataPump USBD contains all the functionality needed to operate the USB bus, including translating USBDI requests into the simpler commands used by host controller drivers, pipe management, default pipe management, bandwidth allocation (for USB 2 host controllers), abstract scheduling for periodic traffic, and class driver management and matching. It adheres closely to the concepts set forward in chapter 10 of the USB 2.0 specification.

Host Controller Drivers

Host Controller Drivers (HCDs) provide a common, portable, low-level API that is used by the USBD to access the physical Host Controller Interface (HCI). Like the DCD API, the HCD API is optimized for high throughput, zero copy DMA operations.

The DataPump HCD architecture has several unusual features. HCDs can run "stand alone", without a USBD. MCCI's "HCDVT" tool uses this feature to perform unit-testing of HCDs and HCIs without the limitations imposed by USBD.

Notable HCDs include:

- Synopsys DesignWare USB 2.0 IP. This HCD supports the Synopsys DesignWare high-speed USB host IP running in host mode, either as part of an OTG core or in a fixed host configuration. Depending on the HCI version, PIO, buffer DMA and scatter-gather DMA are supported. Transaction Translators are fully supported, to allow use of low- and full-speed devices behind high-speed hubs. High-bandwidth isochronous and interrupt pipes are fully supported, which allows use of standard PC webcams. It also includes support for the DWC_OTG core configured as a High-Speed Inter-Chip (HSIC) USB host.
- Mentor Inventra IP. This HCD supports the Mentor Inventra core running as a dedicated USB host, or as part of an OTG device. PIO and external DMA are supported.
- Renesas '597. This HCD supports the Renesas '597 running as a dedicated host, or as an OTG device in host mode. It also supports the equivalent Renesas IP as part of an SOC. PIO or external DMA is supported.
- xHCI. This HCD supports any host controller conforming to the xHCI 0.96, 1.0 or 1.1 specifications. It has been qualified with Renesas, Fresco Logic, TI, Asmedia, Marvell, and Synopsys DesignWare cores.

A key part of the HCD architecture is the "HCDkit" library. This library provides a framework for "typical" host controller interfaces, including such functionality as a simulated root hub device and routines for managing scheduling of periodic transfers for HCIs that require software assistance.

DataPump Host Class Drivers

MCCI provides the following class drivers for the host stack.

- Hub driver – this is the only mandatory driver for the system. In restricted resource environments, the maximum hub depth can be pre-configured. This driver is normally configured to support transaction translators, however in full-speed only systems or HSIC systems with no embedded transaction translators, this support may be omitted.
- Composite driver – supports composite devices by dividing the device up into multiple virtual functions, which then match regular class drivers. This is only used when the host stack is configured as a native or hybrid stack.
- Mass Storage – these drivers support normal mass storage devices, for use when the host stack is configured as a native or hybrid stack.
- HID keyboard and mouse
- Abstract NIC (ECM, NCM, EEM)
- Generic Driver
- Null Driver (for OS emulation)

USB On–The-Go Support

The DataPump host and device stacks have been validated to support USB OTG 1.3. They are believed to be ready for OTG 2.0, but have not been validated.

Common Libraries

The common libraries for the DataPump environment include a rich set of primitives that simplify development.

- The DataPump object system provides a consistent behavior for structures that are registered with the system. Objects are named and discoverable. A standard message system allows objects to implement, inherit, and delegate abstract messaging services.
- Abstract memory allocators model memory pools (which may be implemented by the operating system or by library code in the DataPump working with a fixed amount of pre-allocated memory). Working with preallocated memory allows for "zero surprise" design; working with the operating system's allocators allows for a variable memory footprint that grows based on usage profile.
- Safe memory and string functions allow for runtime buffer overrun prevention.
- The abstract annunciator system allows multiple clients to register with event producers inside the stack, without being aware of the details of how the events are plumbed.
- A comprehensive debug logging system allows for a variety of approaches to runtime sequence-of-event recording.
- A compact, portable UTF8 library allows handling of Unicode strings in the common situations that arise for USB applications.

Operating System Integrations

MCCI offers a variety of pre-packaged operating system support packages, including:

- os/none – this is an MCCI nano-kernel that provides exactly and only the basic services needed for running the DataPump "on bare iron" without an operating system. It essentially provides an interrupt abstraction layer, hardware initialization services, and an event loop.
- Windows kernel – the DataPump host and device stacks can be embedded into WDM drivers. The host stack is provided with wrappers that completely emulate the standard Windows USB host stack, allowing use of standard Microsoft and third-party class drivers. The device stack uses MCCI proprietary APIs to expose the upper edges of the class protocols to user-mode applications.
- Linux kernel – the DataPump host and device stacks can be embedded into Linux kernel drivers, as loadable modules. The host stack is provided with wrappers that completely emulate the standard Linux USB host stack. The device stack is used with the MCCI DataPump native device class protocol implementations.
- Windows user mode – when running over MCCI Catena products, the device and host stack can be run in user mode. A special driver is used to make registers and interrupts directly available to the user mode application containing the DCD and/or HCD, the DataPump Device Framework, and the class protocols. This is particularly convenient when cross-compiling embedded systems for development on the desktop Windows platform.
- VxWorks – the DataPump is integrated as a task, communicating with clients using a DataPump event queue and VxWorks semaphore.
- Micrium uC/OS III – the DataPump is integrated as a task, communicating with clients using messages.
- ThreadX – the DataPump is integrated as a service task, communicating with clients using shared memory or classic device driver techniques.
- Nucleus – the integration is similar to that for ThreadX.
- MQX – the integration is similar to that for ThreadX
- uItron – the DataPump is integrated as a service task, communicating with clients using messages and event flags.
- OSE – the DataPump is integrated as one or more OSE tasks, which communicate by clients using OSE signals (messages).

MCCI can readily port the DataPump to proprietary environments.

MMUs and multiple address spaces are represented by associating a "handle" with each buffer pointer. Handles are opaque to DataPump code, but are used by the operating system layer to record such things as MDLs (for Windows kernel-mode drivers), or signal buffer pointers (for OSE). Buffers without handles are treated as internal kernel-mode buffers.

Validation Tools

To help MCCI and MCCI customers to validate systems built with the DataPump, MCCI has created a number of software and hardware tools, including:

- HCDVT – verification tool for host controller drivers, used for automated unit test of HCDs and HCIs.
- MBIMDVT – verification tool for MBIM devices, fully implementing the MBIM 1.0 compliance specification.
- MSCDVT – verification tool for mass storage class devices, including high-speed and super speed UAS device verification.
- USBDVT – verification tool for MCCI USBD, used for automated regression test of USBD in systems with limited user interfaces.
- WMCDVT – verification tool for WMC class devices.
- Catena systems, including low-, full-, and high-speed test devices and hosts.
- The MCCI USB 3.0 Connection Exerciser, used for automated plug/unplug testing of devices.

Build System and Build Tools

The DataPump is delivered with a complete build system that works on Windows, Mac OS X, Linux, NetBSD, and Solaris. The build system includes the following tools.

- "mccimake" is a version of BSD make, as enhanced by MCCI for cross-platform Makefile portability
- "usbrc" is the USB resource compiler. It verifies, maps, and translates high-level descriptions of USB devices into the concrete descriptors and matching endpoint assignments for the DCI being used in the target system.

The build system treats the source tree as "read only", and integrates readily with OS-specific build systems. For example, the Windows version of the USB 3 host stack is built by creating the DataPump libraries, then building drivers using build.exe in the normal way.

Documentation

- User Guides
- Application Notes
- Test and Verification Procedures
- Detailed technical implementation documentation, in the form of CHM files.
- Source code, which averages more than one line of documentation commentary per line of code (as measured by CCCC).

Architectural Overview

The simplest DataPump device architecture is shown in Figure 1. The USB device hardware is modeled as a USB Device Controller Interface (DCI), which is controlled by a Device Controller Driver (DCD). Because USB transceiver control is often quite tricky in embedded systems, the transceiver (PHY) is explicitly modeled by a transceiver API. (Modeling the PHY also allows simple coordination of "car kit", battery charging, and other features that multiplex non-USB signaling over the USB connector.) The DCD is in turn controlled by the DataPump Device Framework, which in principle supplies all the chapter 9 functionality. (Some DCIs provide some of the chapter 9 functionality in hardware; in such cases, the DataPump Device Framework simply shadows the operations that are being performed by the DCI.)

The DCD does not implement any higher-level USB knowledge; that functionality lives in the DataPump Device Framework and the class protocol modules. DataPump Device Framework functions do not change based on the function being performed by the device. Rather, the function and configuration of the stack is driven (at the chapter 9 level) by the descriptors; and the provision of class protocols is driven by the "application initialization tables".

The DataPump programming environment is much like that for the Microsoft NDIS network driver environment, in that all the component modules are consistently written using a set of primitives and APIs that are independent of the native operating system. Included in the basic facilities is a simple object dictionary. This dictionary allows internal protocol instances (and external clients) to dynamically discover objects within the DataPump environment. A general-purpose messaging system allows external clients to send control messages to any of the indicated API points within the DataPump. This allows native OS management facilities (for example) to control the transceiver, to start/stop the USB device subsystem, to register for and to receive notifications from the any of the layers within the DataPump, all without having detailed knowledge of any of the internal data structures of the DataPump, and without having direct access to the memory being used by the DataPump.

Any USB device must have a static set of descriptors. The DataPump allows these descriptors to vary over time in a number of ways, but it requires that the set of device and configuration descriptors be set before beginning device operation. While initializing, the DataPump then builds data structures that model the device's topology, by examining the descriptors. Finally, it scans the descriptors and notifies each of the provisioned protocol modules, which in turn bind themselves to configurations, interfaces and endpoints, using common code provided by the DataPump framework.

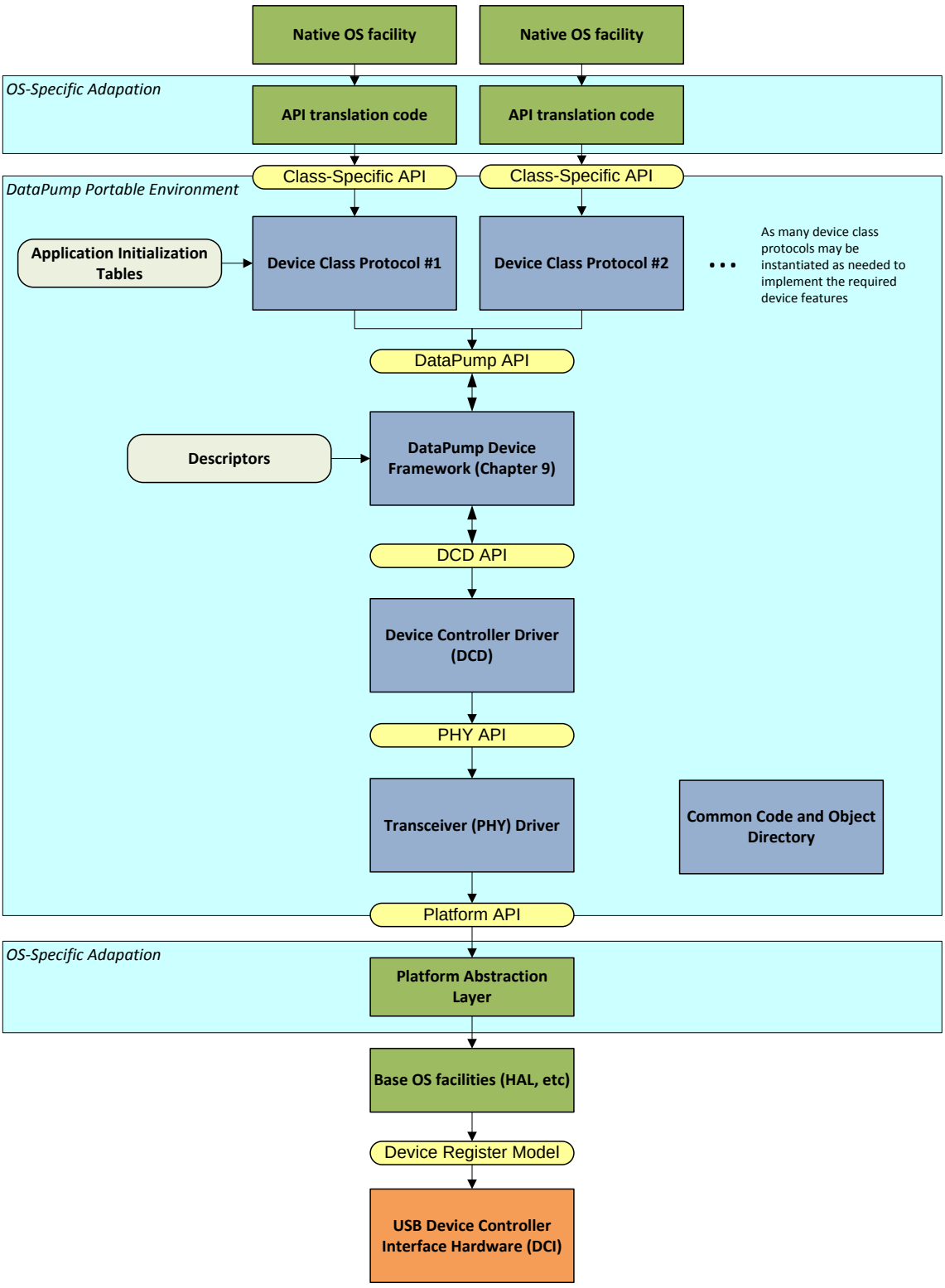

Figure 1. DataPump Device Architecture

(In most cases, the actual "examination of descriptors" is done at compile time by a special tool called "USBRC" – but the architecture of the DataPump also allows this to be done at run time, if desired. Using USBRC allows a much larger number of consistency checks to be done in user mode at compile time, and allows for complex and correct mapping onto the limited hardware found in most SOCs. After parsing the description of the device, USBRC generates the actual descriptors to be used, and the initialization code, as C source code.)

After initialization, device class protocols work primarily with UDATASTREAM structures. These structures model active endpoints, taking into account the possibility of alternate settings and alternate configurations that may allow a logical data path to be established in a number of different ways. For example, a device might have different functionality in full speed mode as compared to high-speed. UDATASTREAMs allow the class protocol modules (and external clients) to deal with devices of arbitrary complexity in an extremely simple and intuitive way.

High-volume data transfer is accomplished in a way similar to that used in USB host stacks. The native client prepares a buffer, or a scatter gather list, and passes it to the adaptation layer. The adaptation layer or the class protocol prepare a USB data request (called a UBUFQE), which is then submitted for asynchronous processing. When the request is completed, a client-supplied callback function is invoked to complete processing. The structure and APIs are optimized for DMA-based DCIs. Because of the short code paths, and the copy-free architecture, data transfers normally operate at bus speed.

The native operating system is modeled through an abstraction layer. This layer has two primary responsibilities: to model interrupts for the DCD, and to provide an "event dispatch" facility. In addition, it provides bindings for the normal OS services (such as allocating and freeing memory).

The DataPump is essentially singly threaded, although it operates well in symmetric multiprocessing (SMP) environments such as the Windows kernel. The primary justification for this is that USB devices, at the USB level, have limited concurrency available. Since DataPump computations are event-driven and asynchronous, rather than a blocking and synchronous, the effect is that if a second CPU offers load to the USB while the first CPU is working, the work is simply queued, and the second CPU is immediately released for other work.

Figure 2 shows a much more complex system. In this implementation, the DataPump is configured to support USB On-The-Go. In addition, the system is configured to use DataPump native class drivers. (It's possible to configure the DataPump to use native OS class drivers; this is how MCCI's USB 3 host stack for Windows is implemented. A detailed description of the architecture is beyond the scope of this introduction.)

The diagram has been slightly reorganized compared to Figure 1, to make it more concrete. We show the DataPump configured for WMC and MSC device functions (either as a composite or

multi-configuration device); and the host stack configured to support HID and Mass Storage class devices.

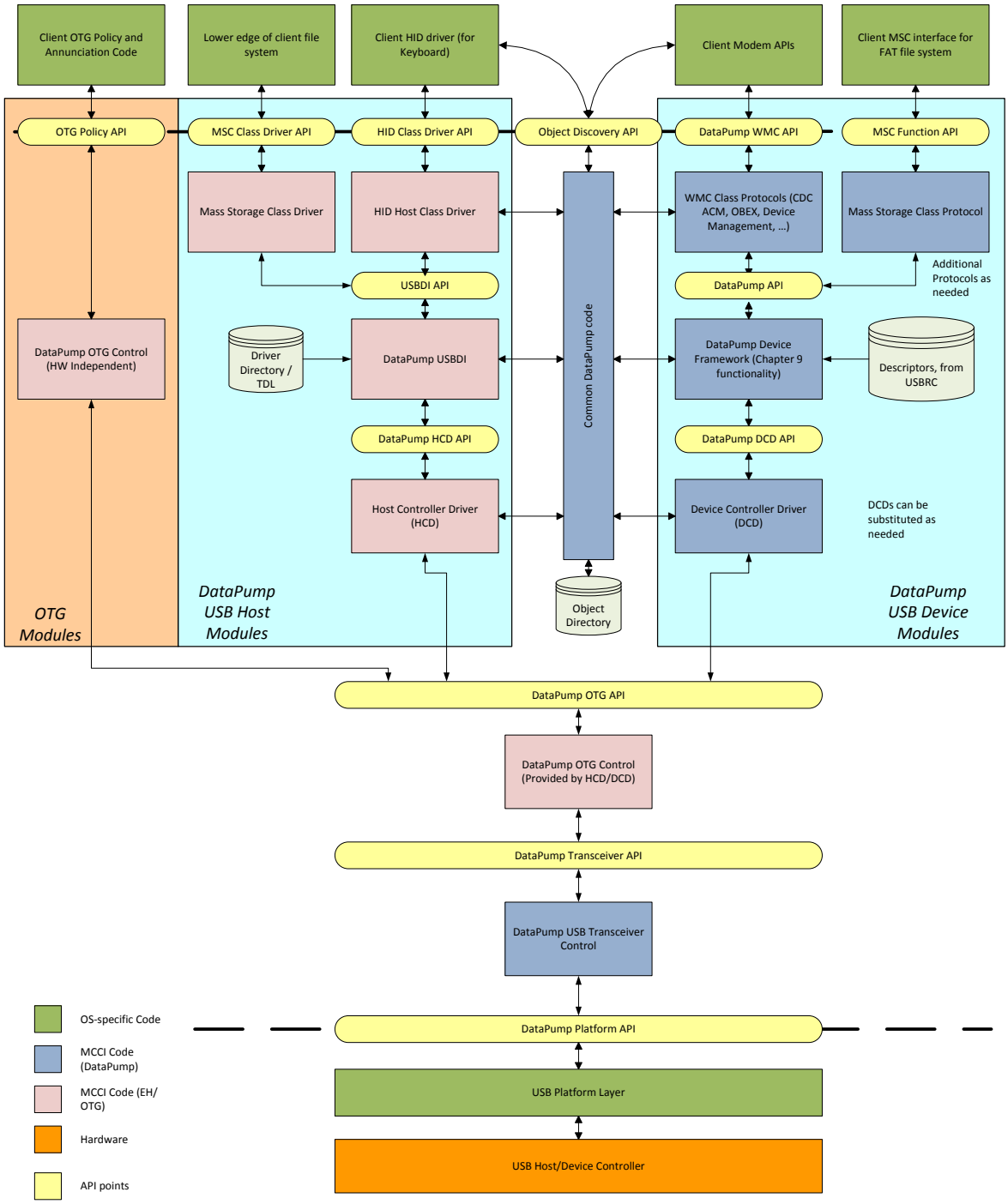

Figure 2. DataPump OTG Architecture

DataPump Coding Conventions

MCCI's coding style has a number of features that contribute to portability.

- ANSI C 89
- No use of global or static variables (global and static constants are, of course, allowed)
- No conditional compiles in C files (except for the usual checked/free configuration)
- Dynamic initialization and configuration
- Library-based
- Abstract types for portability
- Avoidance of type casts
- Type cloaking allows DataPump code to be integrated cleanly with any operating system.
- API versioning discipline provides stable APIs. Ensures strict compatibility for client source code as APIs evolve to meet new requirements.
- Minimal duplication of code

Licensing and re-deployment options

The DataPump product was designed as an OEM ("white label") software package, to allow our customers the most flexibility. It can be licensed in a number of forms.

A full source license is available, which gives licensees the most flexibility in adapting the DataPump to their needs. This approach may also simplify debugging.

Because the DataPump is configured at link time, the DataPump can also be licensed as header files plus libraries. The DataPump is not configured at compile time, so this license gives full functionality. Source licensees who need to sublicense the DataPump can also take advantage of this mode of operation for their sublicensees.

In certain configurations, the DataPump can be delivered as a fully pre-compiled executable. For example, when used as a Windows-compatible host stack, the DataPump is fully compiled and pre-configured. The DataPump device stack can be similarly pre-configured.

Modes of Integration

When integrating the DataPump into an existing system, MCCI offers the following approaches:

Native stack

Configured in this mode, the DataPump serves as a complete USB subsystem. Clients use the MCCI class protocols and APIs directly. This approach is the most efficient.

Emulation stack

Configured in this mode, the DataPump provides core USB functionality, but class protocols are implemented by client code outside the DataPump, typically enabling existing OS-native class drivers to be used without changes. The most typical example of this is using the DataPump in the Windows kernel for USB 3.0 host support, or using the DataPump on Android systems to provide carrier-grade USB support while emulating the existing Linux device and host APIs, so the operating system's broad range of class drivers can be used.

Hybrid

When configured as a hybrid stack, the DataPump uses MCCI class protocol and class driver modules for certain key functions, and uses client code for the remaining functions. An example might be to use the MCCI NCM class driver for high performance zero-copy scatter/gather support, and to use OS-native drivers for other, less performance-critical functions.

Special Applications

Automatic Driver Installation

The DataPump provides complete support for automatic driver installation on Windows and Mac OS X. The device initially is configured as a virtual CD-ROM device, allowing the user to quickly install drivers from a CD image located on the device. After driver installation, MCCI or third party drivers select an alternate configuration of the device; the alternate configuration need not contain a mass storage device.

NCM support (Abstract NIC API)

MCCI's support for NCM is extensive. The API upper edge allows zero copy, gather on write for very fast throughput. In addition, the same API is exported by NCM host and device implementations. The same API can also be used for EEM, ECM and RNDIS (device only) support.

USB UICC support

MCCI's EEM, MSC BOT, Composite, and Generic drivers provide a complete solution for implementing USB UICC. Special PHY code handles the special voltage switching needed during enumeration.

Systems using USB UICC normally will also support high-speed USB device and host. The reentrant pure-code style of the DataPump make it possible to reuse the same code for all these USB ports (running perhaps in a separate thread or with a different data address space, for security) – code and cache footprint is thereby minimized, and support costs are reduced.

HSIC Test (Catena 1910)

The MCCI Catena 1910 uses the DataPump, configured as a Windows host stack, to operate the Synopsys DWC_OTG IP, configured as an HSIC host controller. The only differences between this stack and the MCCI Windows USB 3.0 host stack for xHCI are the HCD, and the link-time configuration.

MirrorLink™

MCCI's high-performance NCM support, combined with the DataPump's flexible implementation style, makes it easy to implement MirrorLink hosts and devices. The MCCI Catena 2210 is the industry standard test fixture for testing USB hosts that implement MirrorLink.

About MCCI

MCCI (www.mcci.com) was founded in 1995, focuses on connectivity software for electronics manufacturers. In addition to the MCCI USB DataPump, MCCI offers a comprehensive range of host class drivers for Windows, OS X, Android, Windows Embedded, and QNX. MCCI's test lab offers Certified USB logo testing and Windows DTM testing services. MCCI also manufactures specialized test equipment for USB and HSIC USB development.