THE ON-THE-GO SUPPLEMENT TO RELEASE VERSION 2.0
OF THE USB STANDARD OPENS THE DOOR TO NEW
CONSUMER PRODUCTS THAT CAN EXCHANGE DATA
WITHOUT CONNECTING TO A PC HOST.

USB OTG software frees
dual-role handheld devices

The new OTG (On-The-Go) supplement to release Version 2.0 of the USB standard allows devices to exchange data without an intermediate host. OTG establishes a new USB product category—dual-role devices that can function as regular USB peripherals or as limited hosts, directly controlling data exchanges. Hosting is intrinsic to USB. When you connect two OTG dual-role products, the end of the cable that initially connects each device determines their roles. However, these roles can reverse via OTG protocols in a manner that is transparent to the end user. Essentially, issuing a file-transfer on either dual-role device makes that device the host.

An example scenario is Maria, a teen who uses her OTG-enabled smart phone with a variety of USB and USB-OTG devices (Figure 1). Beginning the day in her classroom, she uses a USB keyboard as a peripheral and her phone as a host to type an e-mail. At home, she synchronizes with her PC, a standard-USB full-time host, and downloads a new MP3 track into her phone, acting as a peripheral, before going to a party. As she is leaving, Maria plugs her USB headset, which implements only an OTG peripheral role, into her phone, again in a host role, and calls her friends. While riding to the party, she plugs her phone into the car’s jukebox system (host) to play her music play list. At the party, Maria and her friend Lisa connect their cell phones to share address-book entries. Lisa then uses her OTG-enabled camera to load photos onto Maria’s phone. Later, at home, Maria connects her phone to her USB photo printer and prints her favorite picture.

DUAL-ROLE USB DEVICES

The OTG supplement to the USB standard became necessary because USB’s developers originally envisioned it exclusively as a conventional peripheral bus. USB works well for peripherals that interface only with desktop computers, but it does not permit users to share data, such as photos or sound recordings, without a desktop computer acting as a host controller. A host controller must provide storage for a large number of device drivers; be able to source a large current; and accommodate the familiar, skinny, rectangular USB series “A” host-connector receptacle. Peripherals lack these capabilities and use the squarish USB series “B” connector.

The OTG supplement to the USB 2.0 specification adds new protocols, an alternative way to maintain drivers, backward-compatible new connectors, a reduced drive-current requirement, and a feature called “No Silent Failures.” It allows a device to take on the role of a limited USB host, without the burden of supporting all the functions of a PC host. Instead, a dual-role device needs to provide only a limited host capability, peripheral support, and support for full-speed data transfers. Support for low- and high-speed data transfers is optional. In discussing dual-role OTG devices, the OTG supplement generally refers to “A devices” and “B devices.” The A device is the default host that must supply power to the bus; the B device is the default peripheral and can become the limited host.

The low-level OTG driver, which typically resides in logic in the OTG chip, must understand and deal with an OTG-specific SRP (Session Request Protocol) and HNP (Host Negotiation Protocol). The SRP allows B devices to request the attached A device to...
turn on VBUS, the bus-power wire, and begin a session (see sidebar “The USB interface”). This protocol allows the A device, which may be battery-operated, to conserve power by turning VBUS off when there is no bus activity but still provides a way for the B device to initiate bus activity. The HNP allows a user to transfer the host function between two dual-role devices, so he or she doesn’t have to switch the cable connections. The dual-role A device typically initiates HNP in response to input from a user or a dual-role B-device application.

Together, SRP and HNP allow either device to initiate data transfers. End users need not care which end of the cable they plug into which device, but the electrical characteristics of the cable do establish which device is initially the A device and which is the B device. It doesn’t matter whether the user pushes the file-transfer button on the A device or the B device. If the A device initiates a data transfer, that device controls the session; SRP and HNP are not invoked, because the A device is already the default host. If the B device initiates a data transfer, it might use SRP to request the A device to power the bus; the A device would then use HNP to negotiate a role reversal.

Because dual-role devices cannot economically duplicate a conventional USB host’s capacity to store an unlimited number of drivers on a hard disk, they will incorporate a targeted peripheral list that contains only those devices they intend to share data. A targeted peripheral list may allow manufacturers to identify each peripheral by its type and model number, or it may contain industry-defined OTG peripheral types, such as “OTG keyboard” or “OTG mass-storage device.”

All OTG dual-role products share a common connector known as a “mini-AB receptacle” that can accept either a Type-A or a Type-B mini-USB plug, so that users need not worry which end of the cable they plug into their dual-role devices (Figure 2). When two dual-role devices are connected, the one that gets the mini-A plug is the default host.

The host role can change via OTG protocols. Because OTG devices are typically battery-powered, drive current for a dual-role device can be as low as 8 mA. (Conventional USB hosts and hubs supply 150 mA or more.) The bus-power voltage remains at a nominal 5V.

**NEW FUNCTION WITHOUT OBsolescence**

One important requirement that the OTG supplement spells out is that the dual-role device must have a means for communicating messages to the user. Your application-layer software must be able to detect what’s going on over the interface and inform the user. The OTG supplement calls this required feature No Silent Failures. When the cabling allows you to connect USB devices, but the devices do not support the type of communication you are requesting, the devices will alert you, allowing you to correct the problem. The supplement cites an example in which a B device generates SRP, and the A device attempts to give it control by enabling HNP. In the example, the B device cannot enable HNP, a condition the A device can detect, because the B device operates only at low speed; in host mode, a dual-role OTG device must support USB full-speed transfers—12 Mbps, at a minimum. In this situation, the A device must tell the user that it cannot support the B device.

One of the great virtues of the original USB concept is USB’s ability to handle a large number of peripherals via hubs. To avoid complicating hub design and possibly making existing hubs obsolete, the committee that drafted the OTG supplement elected not to require support for dual-role operations, essentially SRP and HNP, through USB hubs. However, when connected to downstream ports, dual-role devices can operate on hubs purely as peripherals.

Unlike non-OTG USB devices, a dual-role device has both host and device software stacks (Figure 3). The OTG driver decides which stack to use, depending first on which end of the cable is inserted into the device’s receptacle, and then on whether HNP has superseded that configuration. The USB host stack is active when the dual-role device acts as a limited host that is responsible for sending the USB data to the hardware on an endpoint basis. It is also responsible for the “USB driver,” which enumerates the
connected USB devices and maintains their information and “host-class drivers.” The number of host-class drivers is limited and depends on what you choose to provide in the targeted peripheral list. The silicon vendor supplies some of the software in the stacks and the OTG and may supply generic-class drivers that you can modify to accommodate nongeneric targets. When a dual-role device is acting as a peripheral, the USB device stack is active. This stack contains a “device-controller driver” that interfaces with the hardware on an endpoint basis for data transfer over the cable. The stack also implements a USB protocol layer that handles USB protocol specifics and a “device-class driver.” The device-class driver depends on the peripheral function of the dual-role device (for example, as a camera, storage device, or printer).

The OTG driver coordinates the switching of the dual-role device as host and peripheral based on the application request or on a hardware indication. The driver requests the hardware for bus grant/release upon request from application, based on the OTG state-machine state. It is responsible for enabling or disabling the host and device stacks based on the dual-role device’s current function. The OTG driver indicates the results of OTG operations, such as “remote device not supporting HNP” or “OTG functionality is available on alternate port,” and it executes the appropriate actions in case of bus-error conditions.

The application layer communicates with the OTG driver for data-session opening and closing and with the host and device stacks for data transfer during a data session. All of these software components work together for both data transfer scenarios when either the A device or the B device initiates a data transfer (Figure 4).

ONGOING CHALLENGES

A number of semiconductor manufacturers will be offering USB-OTG chips, plus low-level drivers that will handle the lower communication layers, but it will be up to you to adapt your application-layer software to make the device OTG-aware. One of the most pressing challenges has been how to accommodate the need for device drivers with the limited computing capacities inherent in portable devices. Without drivers, OTG peripherals are useless. Consider an OTG dual-role digital

THE USB INTERFACE

According to Release 2.0 of the USB standard, the bus transfers signal and power over a four-wire cable, and the USB-OTG cable adds the ID pins to a standard USB cable (Figure A). The VBUS and GND wires deliver 5V power to devices. Signaling occurs over two differential wires at high speed (480 Mbps), full speed (12 Mbps), or low speed (1.5 Mbps). Low speed supports a limited number of device types, such as mice. The device encodes a clock along with the data. Biased terminations at each end of the cable guarantee input-voltage levels and termination impedances, making it possible to detect when a device is attached at each port and to differentiate between high-, full-, and low-speed devices.

The USB is a host-controlled, polled bus (Figure B). Connecting a host to a device requires interaction between multiple layers and entities, such as the bus-interface layer, which provides physical/signaling/packet connectivity between the host and a device, and the device layer, which provides a view of the device for the system software. A function layer gives the host additional capabilities through a matched-client-software layer. Finally, device and function layers each have a view of logical communication within the layer that uses the bus-interface layer to accomplish data transfer.

A data transfer between a source and an endpoint on a device is known as a “pipe.” Two types of pipes exist: stream and message. Stream data has no USB-defined structure, but message data does. Pipes are characterized by data bandwidth, transfer-service type, endpoint directionality, and buffer size. Powered devices configure a default control pipe that is a message pipe and provide access to the device’s configuration, status, and control information.

Transaction scheduling allows flow control for some stream pipes and employs a NAK (negatively acknowledged) handshake to throttle the data rate and prevent buffers from underrun or overrun situations. When a transaction is “NAKed,” the device retries it when bus time is available.

The flow-control mechanism allows flexible schedules that accommodate concurrent servicing of a heterogeneous mix of stream pipes to allow servicing for multiple stream pipes at different intervals and packet sizes.
camera. It can directly print a picture on an OTG printer, provided that the camera has the right driver. Because the digital camera cannot contain drivers for all OTG printers, it can use only the OTG printers listed in the camera's targeted peripheral list. The same is true if the camera deals with OTG mass-storage devices, such as a compact-flash card or a Zip drive. The camera can use only the OTG mass-storage devices listed in its targeted peripheral list.

The OTG standards body has been addressing these challenges. Defining and using OTG-type device classes is one proposal for resolving these challenges. For example, for OTG mass-storage devices, a device-class specification would describe attributes—such as command set, transport protocol, and file system—of an OTG mass-storage device class. In this way, an OTG dual-role device acting as a limited host would need only one driver for all OTG mass-storage devices compliant with the device-class specification.

The challenge in defining a mass-storage-class specification is picking the right file system. Issues include the size of code required to implement the file system, capacity limitations, and interoperability. If a brand-new OTG file system were required, it would likely be one that a desktop computer would support, because most users want to connect OTG devices to their desktop computers.

In the case of OTG printers, the biggest challenge in defining the OTG-printer device-class specification has been getting printer manufacturers to agree on one common print-control language. The situation involves the perennial open-systems-versus-market-differentiation debate. Manufacturers are still weighing the trade-offs and, as before, need to balance self-interest and the needs of the general public.

Author's biography
Srinivas Yarra is a senior software engineer with Philips Semiconductors Division (San Jose, CA). He is responsible for software development and drivers for USB-host controllers and USB-OTG controllers in PC and embedded environments. He holds a Masters Tech degree in computer engineering from IIT (Kharagpur, India) and a Bachelors Tech degree in Electronics and Communication Engineering from JNTU (Hyderabad, India). You can reach him at srinivas.yarra@philips.com.

These diagrams illustrate the data-transfer sequence as initiated from an “A device” (a) and a “B device” (b).