Managing Bluetooth Profiles: A Billion Served

Greg Burns, CTO, Open Interface North America

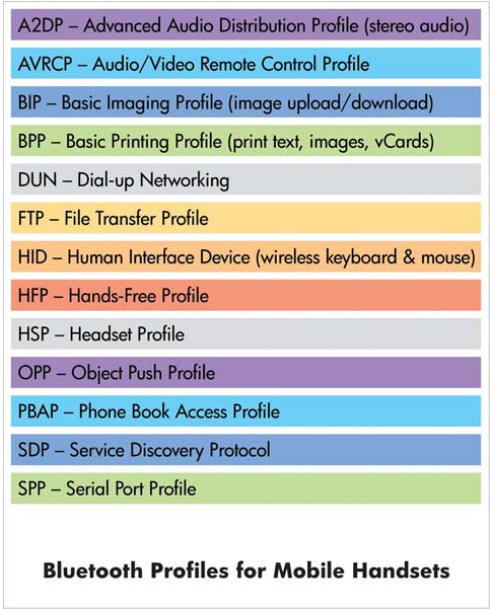
In January 2007, Bluetooth SIG, the member-supported trade organization that defines and enforces Bluetooth wireless technology standards, celebrated a major milestone -- total worldwide shipment of Bluetooth chips surpassed the one-billion-unit mark. Since to make Bluetooth work there must be a Bluetooth protocol stack either embedded on the Bluetooth chip or running on a host processor that communicates with the Bluetooth chip, this milestone also celebrates the shipment of over a billion Bluetooth protocol stacks.

A Bluetooth protocol stack is a software program that sits between a Bluetooth application and the Bluetooth radio. The protocol stack is responsible for implementing the various protocols and profiles defined by the Bluetooth SIG. The protocols specify how different types of control signals and data are packaged and sent between Bluetooth devices. The profiles specify exactly how these protocols are used to implement various types of Bluetooth applications. For example, the headset role of the headset profile defines how a wireless headset communicates with a mobile handset to handle an incoming call, place an outgoing call, and otherwise interacts with a handset. Conversely, the handset role of the headset profile specifies how the handset signals an incoming call and transfers that call to the wireless headset. The Bluetooth SIG operates a compliance program to ensure that Bluetooth devices conform to the requirements of the profile specifications.

These profile specifications are one of the great strengths of Bluetooth wireless technology. They are essentially application-level interoperability standards. Because of the headset profile, a customer can be confident that a headset from manufacturer M will work with a cell phone made by manufacturer N and vice versa. Of course there are important standards at the radio level without which Bluetooth devices would not be able to communicate at all. The Bluetooth SIG enforces all these standards, and only compliant implementations are permitted to exploit the Bluetooth intellectual property and display the Bluetooth logo and brand.

Like the headset profile, most Bluetooth profiles specify two roles: client/server, source/sink, device/host, etc. There are currently specifications for over 50 different profile/role configurations. The table below lists the Bluetooth profiles typically implemented in multimedia mobile handsets. Since a profile defines application-level functions, a protocol stack must provide application programming interfaces (APIs) for each profile and APIs for accessing generic Bluetooth functions that most applications use. To date, the Bluetooth SIG has not undertaken to define standard APIs for Bluetooth, and rather has given free reign to the suppliers to define their own. Some profiles, such as the headset profile, are simple and only need APIs for a handful of functions. In contrast, the basic imaging profile supports still-image upload/download, view-finder preview, printing, archiving, and slideshows, and requires APIs for dozens of functions. All 50 or so profile roles together require APIs for hundreds of functions.

Although the Bluetooth specifications do not cover APIs, there is a specification that defines the bottom layer interface between a Bluetooth protocol stack and the Bluetooth radio. This is the host controller interface (HCI), and support for this interface is mandatory. This means that any well-written Bluetooth protocol stack should be able to work with any Bluetooth radio. In practice, alternative interpretations of the Bluetooth HCI specification can require radio-specific workarounds; but these are minor issues, and generally the HCI standard works extremely well.



In the early days of Bluetooth technology many companies had in-house protocol stack development efforts, and some continue to use their in-house solutions. As the Bluetooth specification has evolved, the cost to develop and maintain a full-featured Bluetooth protocol stack in-house has become prohibitive. Unless cost and time-to-market are not an issue, a developer starting on a Bluetooth project has three potential sources for Bluetooth protocol stack software:

An operating system that has built-in Bluetooth support: Microsoft Windows, Apple’s OSX, various Linux distributions, and Symbian OS include a Bluetooth protocol stack and a limited subset of the Bluetooth profiles

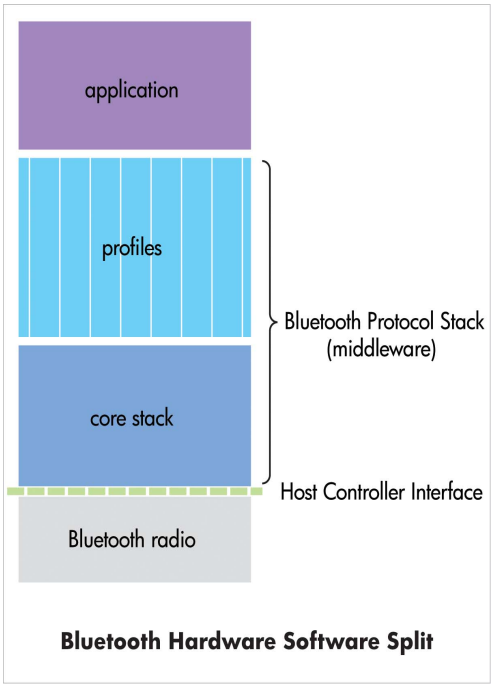
The Bluetooth protocol stack provided with the Bluetooth radio: Most of the major Bluetooth chip manufacturers offer a Bluetooth protocol stack solely for use with their radios.

Licensing a Bluetooth protocol stack from an independent software vendor (ISV): Companies such as Open Interface North America license Bluetooth protocol stack software for specific hardware platforms or can provide portable implementations as source code.

Most embedded operating systems do not have built-in Bluetooth support, so most developers have the choice of using the software provided by the silicon vendor or

software from an ISV.

It may be apparent by now that a Bluetooth protocol stack is a complex body of software. By way of illustration, Open Interface North America's BLUEmagic 3.0 Bluetooth protocol stack is over 100,000 lines of optimized ANSI C source code (excluding header files), and the API set lists over 900 functions. Any full-featured Bluetooth protocol stack will have a similarly rich set of APIs. When choosing a Bluetooth protocol stack, the software designer makes a commitment to become deeply familiar with these APIs, integrating them into the application code and often into audio and network drivers within the product. It is hard to walk away from such an engineering investment.



Bluetooth Hardware Software Split

When starting a Bluetooth project, particularly when adding Bluetooth capabilities to an existing product, the designer must decide which Bluetooth silicon to use. As part of the package, the silicon vendor may bundle a Bluetooth protocol stack with the silicon at little or no incremental cost. This may look attractive, and for one-off products where very limited Bluetooth functionality is required, it may indeed be the most cost-effective approach. But the designer should be aware of the engineering investment being made in that vendor's solution. The standard HCI interface in theory makes it possible for a designer to swap in a different Bluetooth chip as pricing, features or supply dictate. However, if swapping out a Bluetooth chip also means swapping out a Bluetooth protocol stack, the cost of switching chip suppliers may cause product delays, and may require expensive re-testing and re-certification of the product.

Silicon-agnostic Bluetooth software offers significant advantages over using the software provided by the silicon manufacturer. ISVs who focus on delivering portable Bluetooth software platforms can offer greater functionality, more support and better documentation than the silicon vendors whose main business is to sell

chips. For the Bluetooth product designer, in-house expertise developed on one project can be leveraged on successive projects whether or not those projects use the same or different Bluetooth silicon. The hardware designers are free to select the best Bluetooth radio for a specific product, and they can more easily change to a different Bluetooth radio as requirements change. For example, many Bluetooth silicon vendors were slow to deliver production silicon for Bluetooth specification version 2.0. The expense of switching silicon vendors has prevented many companies from upgrading their products to Bluetooth 2.0.

For some companies, particularly the mobile handset manufacturers, the sheer scale of their business requires that they use multiple silicon vendors. Currently this means different products from the same manufacturer using different Bluetooth silicon use different Bluetooth software. With only one or two Bluetooth profiles, this was manageable, but with many devices supporting 10 or more Bluetooth profiles the cost of maintaining different software platforms is becoming an issue. Also, different software from product to product results in inconsistencies between the devices and greatly increases the effort required for interoperability testing.

Even with a billion devices shipped, most of those devices are using a small fraction of the functionality that Bluetooth has to offer. The next billion devices will have more functionality, more complex applications, and correspondingly more points of integration between the applications and the Bluetooth protocol stack. Meanwhile, the cost of Bluetooth silicon continues to drop. Silicon agnostic Bluetooth middleware ensures that designers can build on their software engineering investment and be free to ride the silicon price curve down.

Greg Burns is Chief Technology Officer at Open Interface North America, Inc. an independent software vendor specializing in Bluetooth software and applications. Greg is one of the chief architects of Open Interface's BLUEmagic 3.0 Bluetooth protocol stack. He has more than 30 years experience in embedded software, networking, and consumer electronics. For more information, contact Open Interface North America, Inc., 520 Pike St., Ste. 1770, Seattle, WA 98101; (206) 315-5570; info@openinterface.com [1]*; www.oi-us.com* [2]*.*

Links:
[1] mailto:info@openinterface.com
[2] http://www.oi-us.com/