socket *mobile*

Important *Bluetooth*® and Software Considerations for Wireless Barcode Scanner Deployments

By LEN OTT, Chief Technical Officer, Socket Mobile, Inc.   **February 2011**

Before deploying a *Bluetooth* barcode scanner, you should know the various options available for *Bluetooth* connection types and scanning software, and how these options can affect your business. Most *Bluetooth* barcode scanners today rely by default on a "keyboard wedge" to enter data into a computer, using either a Serial Port Profile (SPP) or Human Interface Device (HID) type of *Bluetooth* connection. Businesses planning to implement *Bluetooth* barcode scanners should understand that (1) SPP provides more efficient data communications than HID as well as support for non-printable characters and (2) using a Software Development Kit (SDK) instead of the default keyboard wedge provides advanced control of many scanner features which can significantly impact the success of your deployment.

History of the Keyboard Wedge

Early barcode scanners were hardwired directly into a computer's keyboard cable, either inline or branching off as a "Y." As a result, the computer would accept the barcode scanner as another keyboard, and both the computer and any applications running on it could not differentiate between information read by the barcode scanner and data typed on the keyboard. This was because each character of data was formatted as a keycode (also known as a scancode), which is the unique code assigned to each key on a keyboard and sent to the computer every time the key is pressed.

This communication approach, known as a "keyboard wedge" or "keyboard emulation," has several advantages: it is easy to implement, requires no special software configuration, and allows users to type in the data in case a barcode cannot be scanned (e.g., due to damage, user error, etc.). However, most keyboard wedges work best only with printable ASCII characters — the characters on a standard English keyboard — plus a few of the non-printable characters such as Tab, Enter, and Page Up. Besides barcode scanners, keyboard wedges are also commonly used with other data collection devices, such as RFID readers and magnetic stripe readers (credit card swipes).

Originally, people connected barcode scanners primarily to desktop PCs. Over time, as people wanted to also connect them to laptop and tablet computers, PDAs, smartphones, and even access points (which lack traditional keyboard cable connectors), manufacturers replaced the keyboard cable with USB or serial RS-232 cables, and later *Bluetooth* wireless technology. Also, keyboard cables

SPP and HID work differently, and not all Bluetooth enabled computers, devices and smartphones support both profiles.

predominantly send data in one direction (from the keyboard to the computer) and many people wanted a two-way cable for scanner configuration.

Although few still connect via a keyboard cable, most barcode scanners continue the legacy of the keyboard wedge and use it by default. Most operating systems (with the notable exceptions of Apple iOS) allow multiple keyboards, so that with a keyboard wedge, users can scan barcodes through a virtual keyboard and type data on a physical keyboard without needing to constantly reconfigure their system or software.

Combining a Keyboard Wedge with *Bluetooth*

There are many kinds of *Bluetooth* connections (known as *Bluetooth* profiles) and the first *Bluetooth* barcode scanners used the Serial Port Profile (SPP), designed to replace serial RS-232 cables between PCs and peripheral devices (e.g., printers). As the *Bluetooth* Human Interface Device (HID) became more popular in the marketplace — mostly for wireless keyboards and mice — many manufacturers started to add support for HID to their *Bluetooth* barcode scanners.

SPP and HID, however, work differently, and not all *Bluetooth* enabled computers, devices and smartphones support both profiles. Additionally, barcode scanner manufacturers do not always provide support for both SPP and HID for a particular OS, even though the OS itself may natively support both.

OS	Native SPP Support	Native HID Support
Windows 7 / Vista / XP	✓	✓
Windows Mobile / CE	✓	✓
Windows Phone	✗	✗
Google Android	✓	✗*
Apple iOS**	✓	✓
Apple Mac OS	✓	✓
BlackBerry OS	✓	✗
Palm OS	✓	✓
Symbian OS	✓	✓
HP webOS	✓	✗

Some Android devices have added HID, but it is not natively supported by the OS.

**Requires proprietary Apple chip.*

SPP vs. HID

If you have the option of choosing between SPP and HID for connecting a barcode scanner to your computer or device, there are multiple factors to consider before making your decision.

If you plan to scan 2D barcodes or even 1D barcodes that contain a lot of data, SPP is inherently better than HID because it communicates data more efficiently. While SPP sends all of the data from a barcode in one packet, HID sends two packets for every *character* in a barcode. For *Bluetooth*, it is much better to send one big block of data instead of many small ones, as the overhead for multiple small packets adds up quickly, leading to slower data communications, decelerated performance for other *Bluetooth* activities, and higher power consumption.

As a result, if you use a scanner in HID mode to read barcodes holding large amounts of data, you may experience a slight delay between when you scan a barcode and when the host computer or device actually receives and displays the data in an application. However, users generally will not notice any speed difference between SPP and HID while scanning typical 1D barcodes with fewer than 20 characters.

Additionally, SPP allows the transmission of non-printable characters (e.g., encrypted data on a driver's license), but HID does not.

Some *Bluetooth* barcode scanners exhibit additional differences between SPP and HID. For example, the Socket *Bluetooth* Cordless Hand Scanner (CHS) Series 7 does not require software installation for HID mode (as is required for using SPP mode), but SPP mode offers many more configuration options, by way of the SocketScan 10 utility.

Feature	CHS with Keyboard Wedge in SPP Mode	CHS with Keyboard Wedge in HID Mode
Software Installation	Required - SocketScan 10 keyboard wedge software	Not required
Barcode scanner configuration method(s)	SocketScan 10 configuration software or special barcodes	Special barcodes
Barcode scanner configuration options	Many configuration options	Limited configuration options
SocketScan 10 software	Supported	Not supported
ActivePairing for roaming between computers / devices	Supported	Not supported

Keyboard Wedge vs. SDK

If your barcode scanner manufacturer offers one, using an SDK to create a custom application is in most cases better than using the default keyboard wedge because it offers more efficient data processing and a broader set of scanner features.

A keyboard wedge translates each character of scanned data first into a keycode (demarcated by "KeyDown" and "KeyUp" steps to indicate when a virtual key was pressed and released) and then hands off the keycodes to the operating

Besides eliminating the need to convert raw data to keycodes and then to ASCII, utilizing a scanner protocol also gives custom applications the ability to integrate a plethora of useful features that keyboard wedges do not offer.

system to be converted into ASCII. Conversely, an SDK understands the specific protocol (proprietary language) of the barcode scanner and presents raw data to the application, eliminating the need for the dual translation steps of a keyboard wedge. Each manufacturer of barcode scanning engines has its own protocol (e.g., SSI for Symbol Technologies / Motorola, ISCP for Intermec). Eliminating the translation steps reduces the amount of data sent over the *Bluetooth* link.

Keyboard Wedge with HID	Keyboard Wedge with SPP	Custom Application with SDK (SPP)
1. User scans barcode. 2. Scanner translates first character of scanned data into keycode. 3. Scanner sends keycode / KeyDown to computer in 1 packet. 4. Scanner sends keycode / KeyUp to computer in 1 packet. 5. Computer converts keycode / KeyUp into ASCII. 6. Computer enters ASCII into software application. 7. Repeat steps 2-6 for all remaining characters of scanned data (including preamble / postamble).	1. User scans barcode. 2. Scanner sends all barcode data (with some standard framing) to computer in 1 packet. 3. Computer translates each character of data (including preamble / postamble) into keycode, with each keycode separated with KeyDown and KeyUp. 4. Computer converts keycodes into ASCII. 5. Computer enters ASCII into software application.	1. User scans barcode. 2. Scanner sends all barcode data to computer in scanner protocol (with some standard framing) in one packet. 3. Computer enters data into software application.

Typically, wireless barcode scanning applications built from an SDK are designed for SPP connections. It does not make sense to use an SDK to create a barcode scanning application based on HID because it is a more complicated protocol than SPP, requires more overhead, and many operating systems / platforms lack the necessary Application Programming Interfaces (APIs) or profile support.

Besides eliminating the need to convert raw data to keycodes and then to ASCII, utilizing a scanner protocol also gives custom applications the ability to integrate a plethora of useful features that keyboard wedges do not offer. For example, the SocketScan 10 SDK enables developers to create applications for the Socket CHS Series 7 that provide automatic data checks and data processing not available with a keyboard wedge. These features are helpful for preventing errors and minimizing user effort while scanning different types of data into complicated applications with multiple fields. Also, the SocketScan 10 SDK enables developers to control scanner features such as symbology support, LEDs, vibrate mode, and beep tones.

For example, a developer of healthcare applications might use the SocketScan 10 SDK to create a bedside medication verification application that incorporates

Feature	Benefit	SocketScan 10 Keyboard Wedge (SPP)	Custom Application with SocketScan 10 SDK (SPP)
Real-time configuration	Better scanner control	✓	✓
Enable / disable symbologies*	Prevent wrong type of barcode from being read	✓	✓
Easy monitoring of *Bluetooth* link*	Advanced users can check status of wireless connection	✓	✓
Non-printable ASCII / non-keyboard characters*	Supports special characters (e.g., null, time stamp, form feed, etc.)	✓	✓
UNICODE, other character sets*	Supports diverse writing systems Almost impossible with typical keyboard wedge	✓	✓
International keyboards*	Supports diverse writing systems	✓	✓
Hide differences between scanner types	Application does not need to know the specific data collection technology or manufacturer	✓	✓
Binary data	Supports barcodes with encrypted data, embedded images or compressed data	✗	✓
Error handling / data checks	Application can verify correct format or value of data for each field Signals user if data is invalid	✗	✓
Data parsing	Application can send data to correct field, even barcodes with data for multiple fields (e.g., address)	✗	✓
Metadata detection and parsing	Application can extract and enter metadata into correct field 2D barcodes often have metadata (e.g., symbology identifiers, GS1 / AIM Global or government defined metadata)	✗	✓
Framing	Application can indicate start and end of barcode data No easy workaround Important for 2D barcodes, which contain more data	✗	✓
Configure start physical scan	For presentation mode or "hands free" scanning	✗	✓
Configure options based on user, application, computer, time of day, last barcode scanned, etc.	Increase efficiency and accuracy of data collection	✗	✓

*Not supported by most keyboard wedges

parsing and data checks to make sure the right patient is given the right medication. To ensure that nurses only scan patient ID numbers from patient wristbands (instead of charts which can have inaccurate barcode labels), a developer can program the patient ID field to only accept patient ID numbers from barcodes that start with a special 2-character prefix found only on patient wristbands. If a valid patient ID is scanned, the application can automatically remove the 2-character prefix and enter only the patient ID . If an invalid or incorrect number is scanned, the application can report the type of error.

Although it is true that someone using a keyboard wedge can enable some advanced features with special programming barcodes, typically very few features can be enabled this way.

Despite the great benefits of using an SDK to create a custom application, not all barcode scanner manufacturers offer SDKs. Some offer SDKs only for a few operating systems and programming languages, thereby restricting some developers to working with certain hardware brands based on their SDK compatibility. SDKs can significantly differ between vendors, resulting in very different implementations of the same application.

Also, using an SDK is not right for every business: if you are not a software developer, and/or only need to scan one type of data into one kind of field, then a keyboard wedge may suffice for your business needs. Moreover, SDKs are difficult to implement for applications using a thin client with processing done "in the cloud" (i.e., any web-based application that utilizes a standard browser). This is because making the application communicate with a barcode scanner through an Internet connection is extremely complicated (if not impossible), and the thin client device can usually only communicate with the application through a web browser or terminal services client.

Conclusion

To maximize the accuracy and efficiency of data collection in your barcode scanning application, it is best in most cases to use the SDK, if available, to develop a custom program that utilizes scanner protocol to enable a wide variety of valuable features. However, if you decide to instead implement a keyboard wedge, and have the option of choosing between a *Bluetooth* HID or SPP connection, SPP will be the better choice if you plan to scan 2D barcodes (or even 1D barcodes holding a lot of data). You should also choose SPP if you plan to scan barcodes with non-printable characters, such as encrypted data on a driver's license.

ABOUT THE AUTHOR

Len Ott is the Chief Technical Officer of Socket Mobile and has been involved with the *Bluetooth* SIG since 2001. He has helped to write the *Bluetooth* Core Specification and Health Device Profile and contributed to the IEEE 802.15 Personal Networking Profile. Mr. Ott is currently a member of the Medical Working Group and a number of the *Bluetooth* Low Energy Working Groups of the *Bluetooth* SIG and also sits on the AIM Global Technical Systems Committee, helping to standardize barcode, RFID, and other data collection standards.