**WirelessUSB™ LS 1-Way HID Networks**

**AN047**

**WirelessUSB LS 1-Way HID Protocol Overview**

WirelessUSB™ LS 1-Way networks utilize a proprietary communication protocol that emphasizes transmitter simplicity and is an ideal low-cost, low-power wireless solution for Human Interface Device (HID). Each HID device contains a WirelessUSB LS transmitter (CYWUSB6932) while a bridge contains a WirelessUSB LS transceiver (CYWUSB6934). The bridge acts like a wired HID device making the wireless connection transparent to the host PC. Therefore there is no special software required on the host PC in order to support WirelessUSB LS. WirelessUSB LS 1-Way networks can achieve a maximum data rate of 64 kbps.

![WirelessUSB LS 1-Way HID System](image)

**Radio Channel Management**

WirelessUSB LS utilizes the unlicensed 2.4-GHz Industrial, Scientific and Medical (ISM) band. WirelessUSB LS splits the band into 79 distinct frequency channels.

**Pseudo-noise Codes**

Pseudo-noise codes (PN Codes) are the codes used to achieved the special matched filter characteristics of DSSS communication. Certain codes referred to as Gold codes are used for WirelessUSB-LS 1-Way communication. These codes have minimal cross-correlation properties, meaning they are less susceptible to interference caused by overlapping transmissions on the same channel. The length of the PN Code results in different communication characteristics which must be weighed when deciding whether to use 32-chip or 64-chip PN Codes. Higher data rates are achieved with 32-chip PN Codes, while 64-chip PN Codes allow longer range.

The number of frequency/code pairs is large enough to comfortably accommodate hundreds of WirelessUSB LS devices in the same physical space. Each bridge/HID pair must use the same PN Code in order to communicate effectively. If there are multiple HIDs in a network, each HID should use a unique PN Code so that overlapping transmissions will not cause data collisions. For more information on frequency division multiple access (FDMA) please read the WirelessUSB LS Theory of Operation white paper.

**Channel Selection Procedure**

For two devices on a wireless network to communicate they must use the same frequency/code pair. The process by which devices are assigned the proper frequency/code pair for network communication is called binding. The LS 1-Way protocol supports two forms of binding, basic and semiautomatic, both of which are described below.

**Basic Bind**

To change channels (initial setup or due to channel interference) the user manually selects the new channel on both devices using a selector switch. The selector switch allows the device to use a subset of the available channels and PN Codes; Bridge/HID pairs should contain identical channel subsets. It is recommended that bridges contain an LED that indicates when messages are received from the HID. In the rare event that a channel becomes bad due to interference, the user can select a new channel on both devices.

**Semiautomatic Bind**

The semiautomatic bind procedure is initiated by user action on both devices, such as pressing a bind button. The procedure differs for HIDs and Bridges, each is described below.
HID
Upon entering bind mode the HID selects a new channel and/or PN Code pseudo-randomly, and builds a bind packet. The HID then switches to the predefined Bind Channel and Bind PN Code and transmits multiple copies of the bind packet, then it switches to the new channel and PN Code specified in the bind packet and transmits a defined number of copies of the ping packet. The HID continues alternating between transmitting the bind packet on the bind channel and PN Code and transmitting a defined number of copies of the ping packet on the new channel and PN Code. The HID then switches from bind mode to data mode; the new channel and PN Code will be used for all data transmissions.[1]

Bridge
Upon entering bind mode the bridge switches to the predefined Bind Channel and Bind PN Code and waits for a bind packet. If a bind packet is received the bridge switches to the new channel and PN Code specified in the bind packet and listens for a ping packet. If a ping packet is received the bridge switches to data mode and listens for data packets. The new channel and PN Code are used during data mode. If the bridge does not receive a ping packet within a defined number of milliseconds it switches back to the Bind Channel and Bind PN Code and listens for another bind packet. If the bridge is in bind mode for a defined number of seconds without receiving a ping packet, the bind failed and the bridge quits listening for bind and ping packets and returns to its previous channel and PN Code.[2]

Data Exchange Procedure
In WirelessUSB LS 1-Way networks there are no ACK/NAK messages and thus no way for the HID to determine if the bridge is correctly receiving transmitted messages. Therefore redundant message transmissions are used to increase the probability of the bridge receiving the message. This is especially important in networks containing multiple HIDs, where time-overlapped messages from multiple HIDs can occur. Each data packet contains a sequence field, which is incremented in every new data packet, making it easy for the bridge to identify and discard redundant copies of data packets.

For 1:1 networks it is recommended that after each data transmission the HID pause for at least the time required to transmit one byte, before sending the next transmission or retransmission. This pause helps the bridge clearly identify the beginning and end of each transmission.

For 2:1 networks all transmissions should use the same frequency channel, with each HID being assigned a specific PN Code to use for data transmissions. When 32-chip PN Codes are used the bridge is capable of listening to both devices simultaneously.
When 64-chip PN Codes are used in 2:1 networks the bridge is not able to simultaneously listen to both devices. Therefore a common PN Code is used by both HIDs to send a preamble before each data transmission. The common PN Code must be different from the PN Codes assigned to the HIDs for data transmissions.

If a HID has data to transmit it will first switch to the common PN Code and transmit its preamble. Then it will switch back to its own PN Code and transmit the data packet. This process is repeated for each transmission and retransmission. The bridge will listen on the common channel for a preamble. When a preamble is received the bridge will switch to the PN Code that matches the HID that sent the preamble and listens for the data message. After receiving the data message the bridge switches back to the common PN Code and listens for the next preamble.
The use of preambles and a common 64-chip PN Code allow the bridge to listen for transmissions from both HIDs simultaneously instead of continuously alternating between the PN Codes assigned to each device. This increases the probability of the bridge receiving either the first or second transmission of a packet, thus decreasing the latency and increasing the actual throughput of the system.

Packet Structures
Little Endian transmission is used in WirelessUSB LS. Where checksums are specified, the checksum is a byte-wise XOR of all preceding bytes.

Bind Packet
Bind packets are used in the semi-automatic bind procedure to specify the new channel and PN Code to be used for future data transmissions. Bind packets contain three bytes as described below:

<table>
<thead>
<tr>
<th>Channel</th>
<th>PN Code</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x5A</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

Ping Packet
Ping packets are used in the semi-automatic procedure for validation of the channel and PN Code specified in the Bind packet. Ping packets contain one byte as described below:

| 0x5A |

Notes
1. Note: The number of packet transmissions is configurable
2. The timeout values are configurable.
Data Packet
Data packets are used in the data exchange procedure for the transmission of application data. Data packets contain at least three bytes as described below:

<table>
<thead>
<tr>
<th>Data Byte 1</th>
<th>Sequence ID</th>
<th>Checksum</th>
</tr>
</thead>
</table>

OR

WirelessUSB LS HID Architecture

HID Hardware Architecture
WirelessUSB LS HIDs use an external microcontroller (such as a Cypress Semiconductor Corporation PSoC) to perform the specific I/O functions necessary for the HID application (i.e., keyboard, mouse or gamepad). The microcontroller also performs all WirelessUSB LS protocol tasks. The microcontroller is responsible for managing low power mode by signaling the radio to enter and exit sleep. User activity is detected by the microcontroller, which then commands the radio to wake up in order to begin sending HID reports to the bridge. HIDs remain in sleep mode between user input events to maximize battery life.

Figure 2. WirelessUSB LS HID Hardware Architecture

Example HID Firmware Architecture
The HID firmware may be customized to fit each specific device. The illustrations and explanations provided in this section show an example of how the WirelessUSB LS 1-Way HID firmware could be designed. The following diagram illustrates the typical interaction of the WirelessUSB LS 1-Way HID tasks for keyboard and mouse applications.

Figure 3. HID Firmware Architecture

Application Process Task
The Application Process Task defines the specific function of the HID. For a keyboard HID, the Application Process Task is responsible for performing keyboard scanning, for a mouse HID, the Application Process Task is responsible for communicating with the optical mouse sensor and for detecting mouse button events, and for a gamepad the Application Process Task is responsible for handling analog to digital conversion and detecting button events. All wireless traffic from the HID is ultimately initiated by the Application Process.

Protocol Task
The Protocol Task is responsible for polling the Application Process Task for events, assembling packets from the Packet Buffer Pool and executing the Data Exchange Procedure. Refer to the Protocol Overview Section for details about this procedure.

Note
3. The number of data bytes per packet is predefined by the application layer.
WirelessUSB-LS Bridge Architecture

Bridge Hardware Architecture
The WirelessUSB LS Bridge contains a USB chip that interfaces to a PC, Laptop, or other USB Host device. The WirelessUSB LS Bridge also contains a transceiver (CYWUSB6934) in order to support wireless communications with the HIDs. The WirelessUSB LS Bridge acts as a translation link between the wireless interface and the USB interface.

Figure 4. WirelessUSB LS Bridge Hardware Architecture

The current design uses the low-cost *enCoRe* USB chip, which has two data endpoint buffers. The *enCoRe* design supports a maximum of four active links so in the event of more than two devices connected to a single bridge, the firmware can share each endpoint buffer between multiple HIDs. Upon connection to a host, the bridge automatically enumerates as the target HID(s).

For a low-speed device, a maximum of eight bytes may be transferred per USB transaction. Additionally, a low-speed device may request no more than one transaction per 10 milliseconds.

The bridge firmware supports USB sleep and remote wake-up. When the PC Host requests the bridge to enter low-power mode, the *enCoRe* will in turn request the radio to enter sleep mode. The *enCoRe* will then periodically command the radio to scan for wake-up events from any active HIDs. If a wake-up event is detected, the *enCoRe* will issue a USB remote wake-up request to the PC Host. When the PC Host instructs the bridge to exit low-power mode, the *enCoRe* commands the radio to exit sleep mode and return to normal operation.

Example Bridge Firmware Architecture
The bridge firmware is responsible for managing the received data from the HIDs. The bridge firmware may be customized to fit each specific device; the illustrations and explanations provided in this section show how the WirelessUSB LS 1-Way bridge firmware could be designed. Figure 5 illustrates the typical interaction of the WirelessUSB LS 1-Way bridge firmware tasks.

Figure 5. Bridge Firmware Architecture

Receive Task
The Receive Task instructs the Protocol Task to receive packets. When a packet is received, the Receive Task adds a receive event to the Event Queue, and includes a reference to the appropriate Packet Buffer used for the packet payload (from the Packet Buffer Pool). If the packet is corrupted, it is simply discarded.

Protocol Task
The Protocol Task is responsible for handling events in the Event Queue, disassembling packets, and executing the Data Exchange Procedure. Refer to the Protocol Overview section for details about this procedure.
**USB Service Task**

The USB Service Task is responsible for servicing the USB Interface between the enCoRe chip and the Host PC. The USB Service Task receives commands from the USB interface, performs the specified action, and then issues a response to the USB Host. The enCoRe’s USB interface connects the WirelessUSB LS Bridge to the Host PC USB bus. The firmware is executed by the 8-bit M8 processor within the enCoRe chip. The enCoRe firmware manages the USB interface with the Host PC as well as the WirelessUSB LS wireless interface to the HID. The enCoRe firmware also provides the functionality that makes Wireless HID devices appear to enumerate as conventional wired USB HID devices in the PC environment. Each WirelessUSB LS HID formats its HID reports to comply with USB HID reports, so there is very little additional work for the enCoRe to perform on the packets before passing them to the PC Host.

In March of 2007, Cypress recataloged all of its Application Notes using a new documentation number and revision code. This new documentation number and revision code (001-xxxxx, beginning with rev. **), located in the footer of the document, will be used in all subsequent revisions.

WirelessUSB and enCoRe are trademarks of Cypress Semiconductor. PSoC is a registered trademark of Cypress Semiconductor Corporation. All other trademarks or registered trademarks referenced herein are the property of their respective owners.