Amp'ed RF
Wireless Technology

Fast Track Your Wireless Project
自由无线 放飞梦想

abSerial User Guide

10 Feb 2015

Contents

1 Preface

The document describes an embedded application that provides a wireless serial cable replacement service using the Bluetooth Serial Port Profile, abSerial.

1.1 Purpose

This document provides guidelines for users of host applications that use the abSerial interface. Such host applications may execute on an embedded microcontroller or other device that will primarily pass a serial data stream to the abSerial interface, which will then be transferred using the Bluetooth Serial Port Profile (SPP).

The abSerial interface supports AT-like attention commands for configuration and control. The *abSerial User Guide* explains the commands and sequences needed to use the abSerial interface as a serial port. For a more detailed discussion on each command, please refer to the *abSerial Reference Guide*.

1.2 Definitions and Acronyms

The following acronyms are used in this document.

Table 1. Definitions and Acronyms

Term	Description/Meaning
AT	Text based command standard commonly used for modems
BD	Bluetooth Device
bps	Bits Per Second
CTS	Clear to Send line (hardware flow control input on UART that allows data transmission)
RTS	Ready to Send line (hardware flow control output on UART that stops receiving data)
RxD	Receive Data line (on UART)
SPP	Serial Port Profile
TxD	Transmit Data line (on UART)
UART	Universal Asynchronous Receiver-Transmitter

2 Overview

2.1 Modes of Operation

The software behavior of the abSerial interface is similar to a Hayes-compatible modem. The application has two modes, a command mode and a bypass mode. In command mode, the host can issue commands for configuration or connection management. Note that the abSerial interface does not support standard Hayes AT command set. Instead, it has commands that utilize a vendor-specific command form.

Once a connection is established, the application automatically switches to bypass mode. In bypass mode, data arriving from the local host will be sent over the Bluetooth link to the remote device. Any data received from the remote device will be transferred to the local host.

While in the bypass mode, the abSerial interface will search for an Escape sequence from the host. If this sequence is found, the abSerial will switch to command mode. This allows commands to be issued again from the local host, but the connection to the remote device will remain.

While in command mode, the abSerial interface will send Events back to the host for commands received. Events from the abSerial interface will also be sent on system reset. Event strings are not sent to the host during bypass mode.

2.2 UART Configuration

The default interface of abSerial uses the main UART, with these settings:

- 115200 bps
- 8 data bits
- no parity
- 1 stop bit

Hardware flow control (RTS/CTS) may be used by the abSerial interface, and is recommended in applications where data losses are not tolerated.

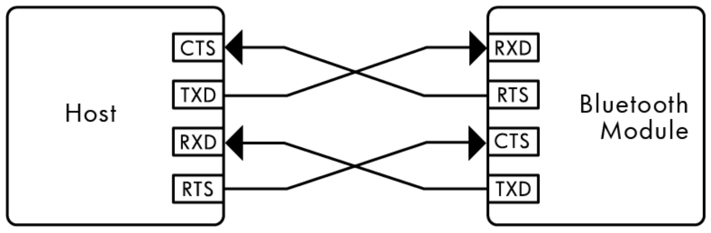

Figure 1. Connection to Host Device

2.3 System Data Flow

The following diagram shows how the data stream into and out of the abSerial interface is a part of an overall system that uses the Bluetooth SPP.

The bytes into and out of the application are of two types. The first is for commands and responses. Commands and responses are handled only while the application is in command mode. When in bypass mode, the second type of data stream is transferred directly to/from the UART and the Bluetooth SPP.

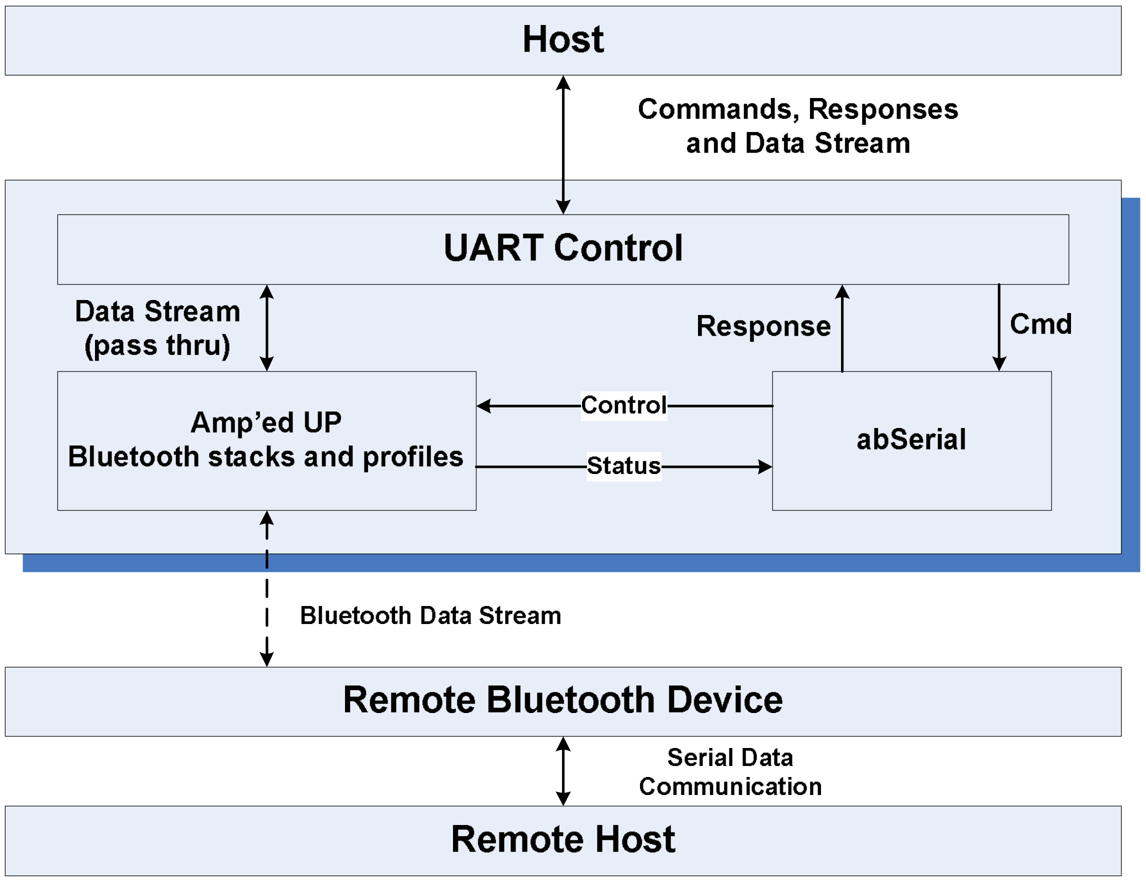

Figure 2. Data Flow between abSerial, Host, and Remote Bluetooth Device

3 Startup

Upon initialization (due to power up or system reset), the abSerial interface starts in command mode and the serial port speed is set to its default baud rate. The application will then send two event strings (if host event strings are enabled). One notifies the host that the abSerial interface is in command mode and the other shows the BD address of the local device.

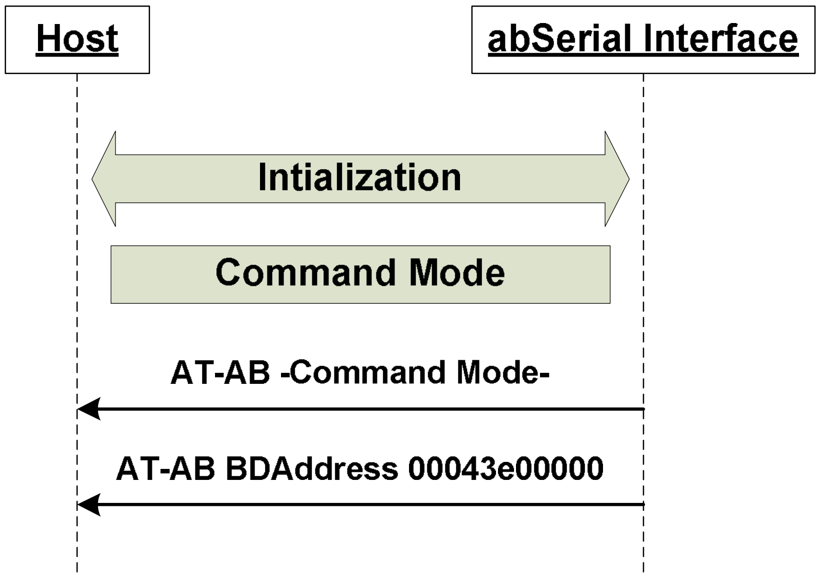

Figure 3. Event Strings Sent on Startup

Then, the application listens for a connection on the SPP profile and remains in command mode until a connection is established.

4 Connect with Remote Device

In order to create a connection with a remote device, the host issues a command string to the abSerial interface. This connection command may only be sent while in the command mode and when there is no active connection. The BD address of the remote device must be known at the time the connection is requested. Once the connection is established, the application goes into bypass mode.

4.1 Successful Connection

If the connection request is successful, the application will go to the bypass mode. The response can take a few seconds.

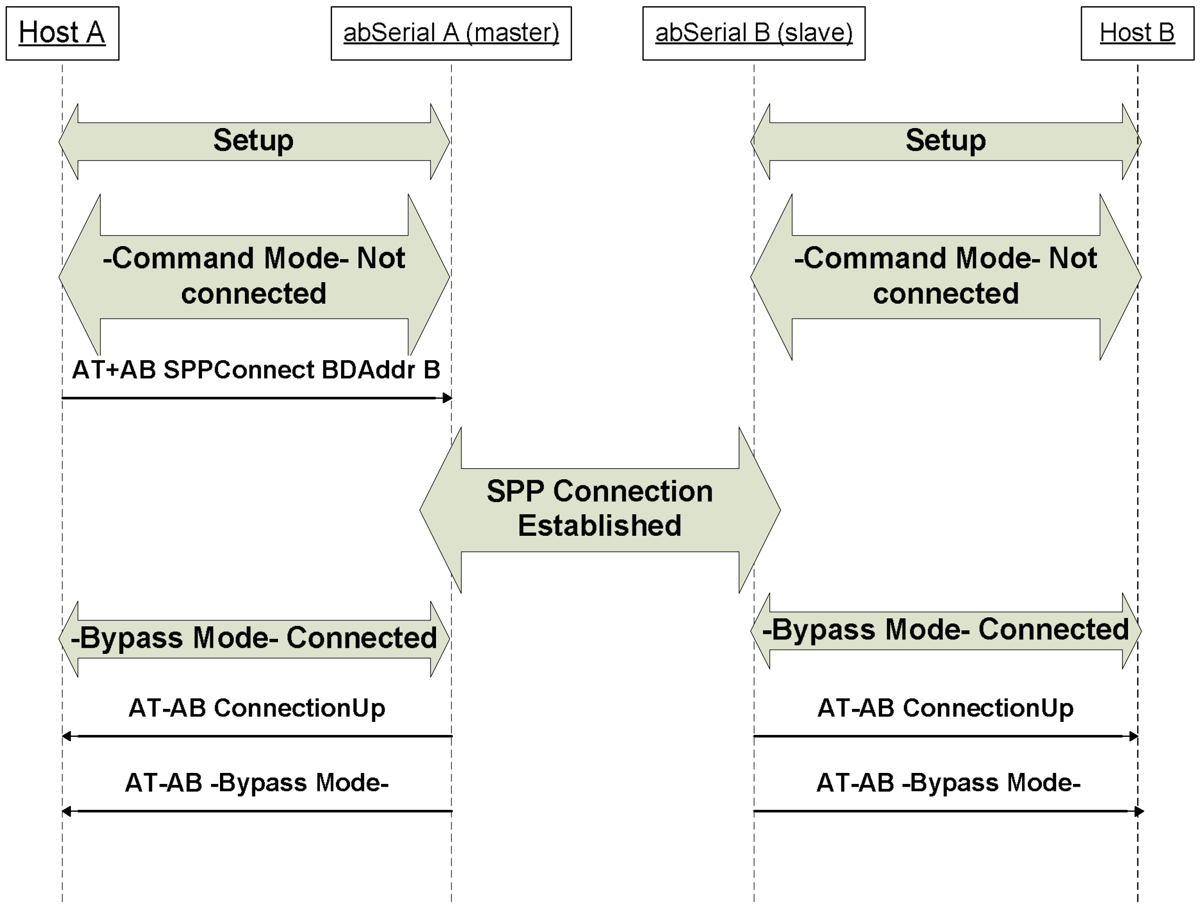

Figure 4. Command and Event Strings for Successful Connection

4.2 Unsuccessful Connection

There are numerous reasons a connection may not be established including remote device rejection due to security or poor RF quality. If the connection request is not successful, the application will remain in command mode. A response can take a few seconds.

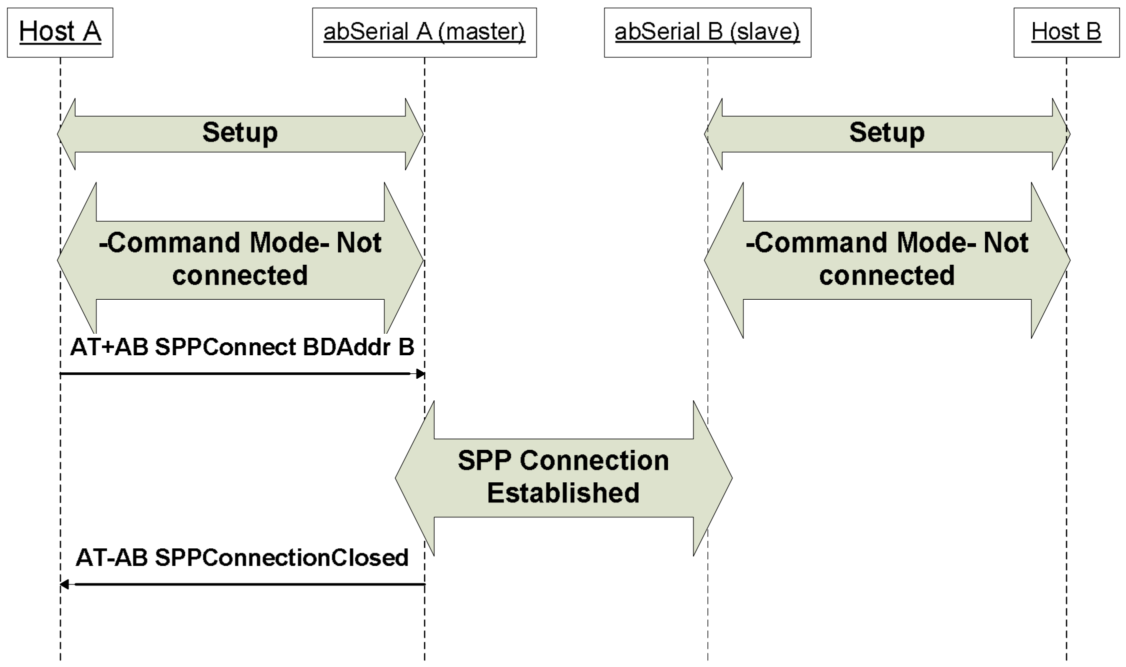
Host A
abSerial A (master)
abSerial B (slave)
Host B
Setup
Setup
-Command Mode- Not connected
-Command Mode- Not connected
AT+AB SPPConnect BDAddr B
SPP Connection Established
AT-AB SPPConnectionClosed

Figure 5. Command and Event Strings for Unsuccessful Connection

5 Disconnect with Remote Device

Once a connection is made, either device may request a disconnect. Also, a disconnect may occur unexpectedly due to changing conditions such as moving a device beyond the reception range. This section will illustrate disconnects under different situations.

In order to terminate an existing connection with a remote device, the host issues a disconnect command string. However, once a connection to a remote device has been established, the application is in bypass mode. Therefore, the host must first put the application in command mode before it can close the connection. The Escape sequence is sent to begin this process. The Escape sequence is discussed in greater detail in Section 6.

Once the abSerial interface is back in command mode, the host sends the Disconnect command string. The application notifies the host when the connection is broken and returns to command mode.

For disconnects initiated due to changing RF conditions, both hosts would receive the same notification as the non-initiating host in the following examples.

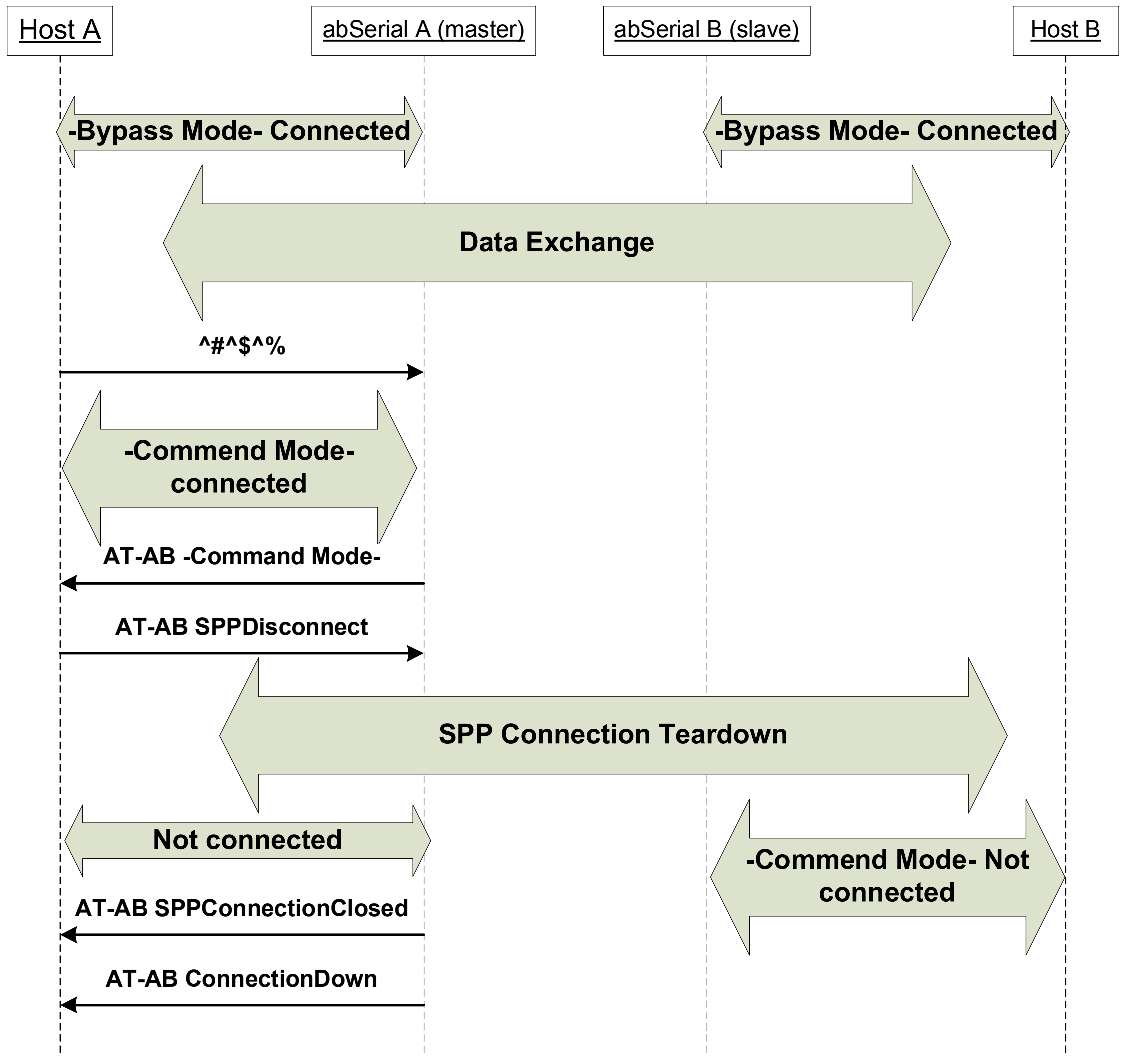

Figure 6. Escape, Command, and Event Strings for Notification Disabled

6 Escape from Bypass Mode

Once a connection has been established between host and remote device, the host can put the abSerial interface back into command mode. Once back in command mode, new commands (including the termination of a connection) can be issued. To change the application out of bypass mode and into the command mode, the Escape sequence is used.

The Escape sequence is an escape string followed by 1000 ms of no data. The Bluetooth connection to a remote device is not affected.

Note: any data received from the remote device while in command mode will be discarded by the local abSerial interface and not passed to the local host.

6.1 Connection Still Active

If the abSerial interface is in bypass mode when the escape string is sent (i.e., the connection is still active), the host must wait 1 second before the application will respond in the command mode.

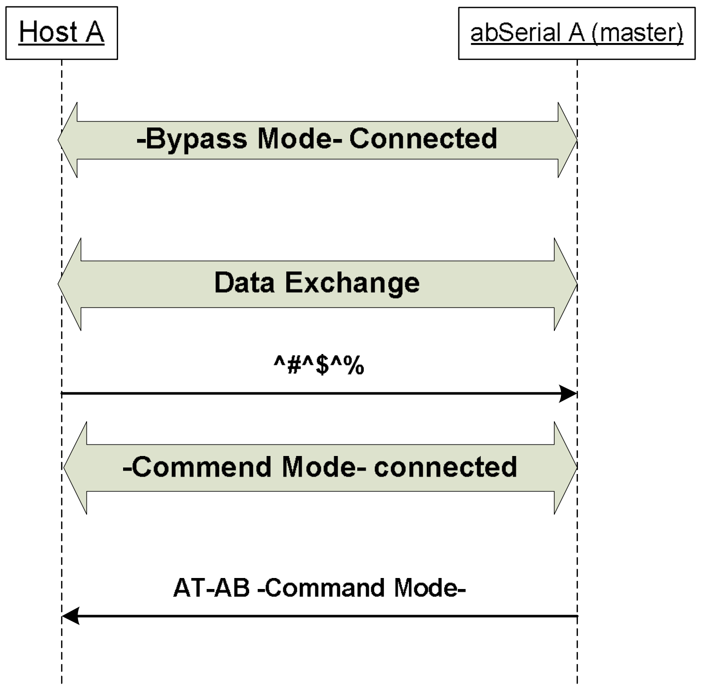

Figure 7. Escape Sequence and Event String when Connection Is Active

Note: the escape string must NOT be followed by a line-feed or carriage return.

7 Device Discovery

A abSerial device can use its discovery command to search for nearby Bluetooth devices. The information parameters returned by the feature to the host are the BD address and remote device name. The abSerial interface can also filter responses to show only particular classes of devices or service profiles. For more information on how to use the discovery command, please refer to the *abSerial Reference Guide*.

On issuing the discovery command, the local device first does inquiry and displays the number of remote devices which responded to the inquiry procedure. The local device then performs a name request procedure on all of the remote devices found in during inquiry, in the order in which they responded. The name request procedure consists of establishing a connection, performing the name request, and then disconnecting.

Once name discovery and service discovery are performed on one device, the same procedure can be repeated for all devices that responded to the global inquiry.

The following example shows a general device discovery that returns two devices.

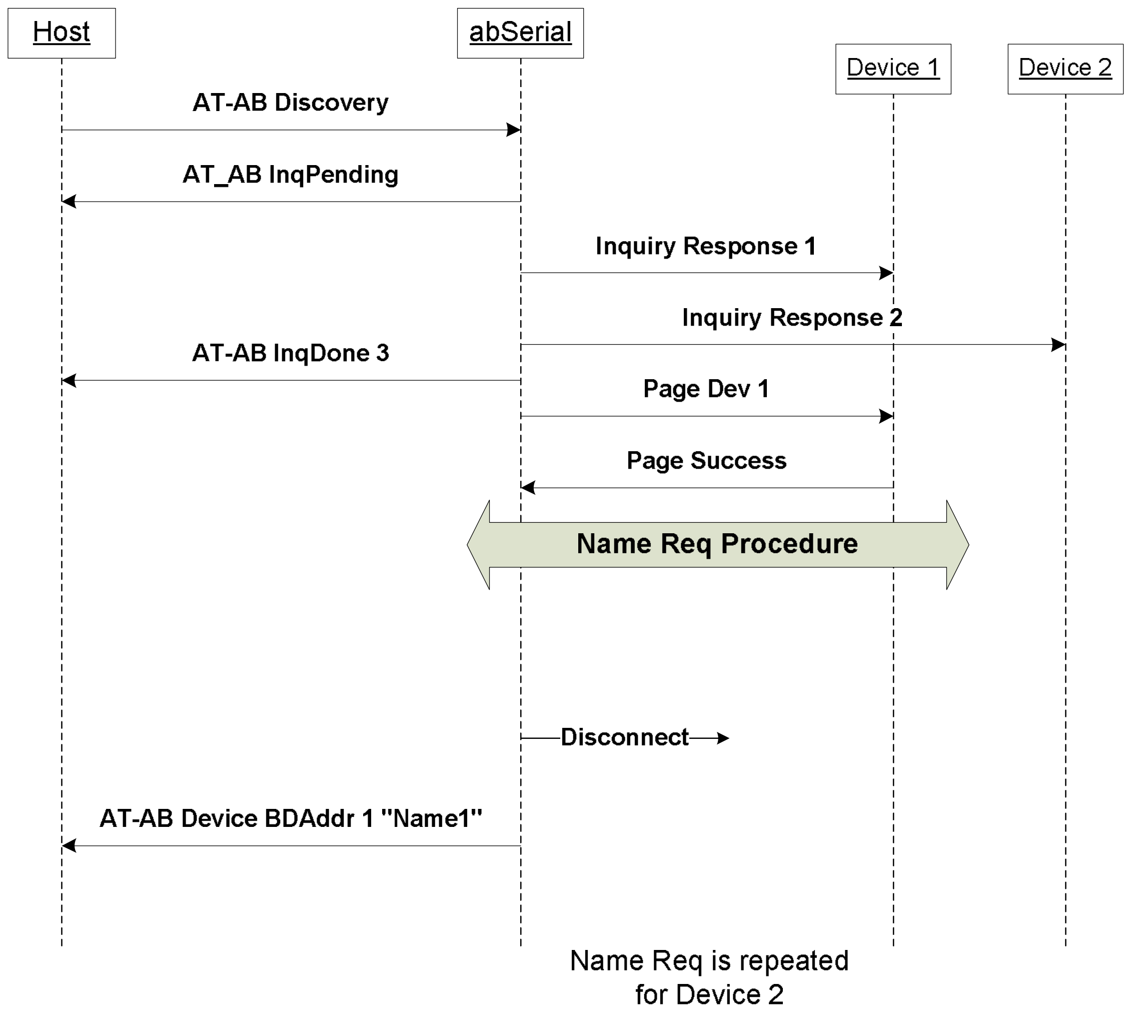

Figure 8. Commands and Events for Device Discovery

8 Bonding and Security

Bonding is used when an application needs to pair with another remote device. When the application issues "`EnableBond`", it allows the local device to accept bonding requests. When the application issues "`Bond`", it causes the local device to initiate bonding with the device to which it connects. Of course, if the application issues "`Disable Bond`" it forbids the local device from accepting any bonding requests, thus not allowing connections from devices that require bonding.

If the application issues "`EnableBond`" and is accepting a connection from a particular device (DeviceA) for the first time, it will:

Initiate authentication with the remote device (also known as Pairing)

Both devices will ask their respective hosts for a PIN which they then send to the other device for verification

If both devices confirm their PIN or PassKey codes, bonding succeeds and an encryption LinkKey is generated.

Store the new LinkKey for Device A.

If one of the devices does not verify the PIN, bonding fails and the connection is terminated.

Upon successive connections to DeviceA, the abSerial device will automatically use the stored LinkKey to authenticate the remote device and initiate encryption without notifying the abSerial host.

The following example shows how the abSerial interface can be used for bonding.

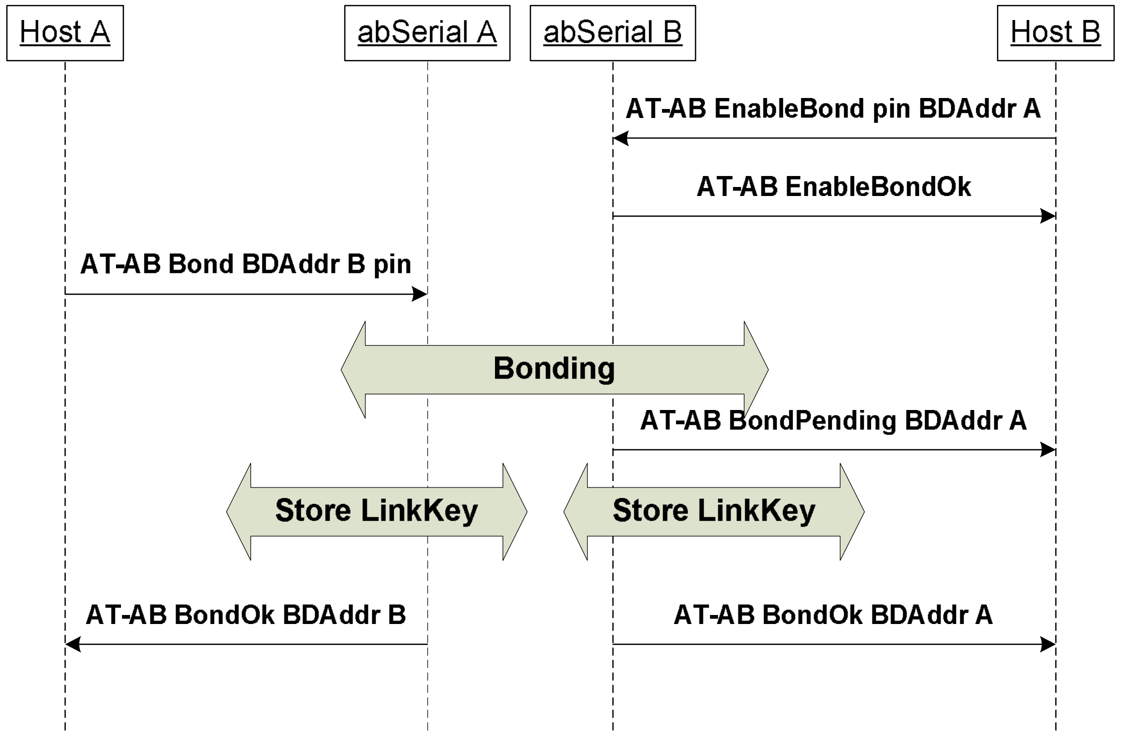

Figure 9. Command and Event Strings for Successful Bonding

If bonding is not successful, then `AT-AB BondFail` is sent to both hosts instead of `AT-AB BondOk`.

For further details on “`Bond`”, “`EnableBond`”, please refer to the *abSerial Reference Guide*.

9 Remote Escape Sequence, "@#@$@%"

The purpose of this feature is to allow a remote abSerial device be controlled and configured by a Bluetooth link using a local host and Bluetooth device.

NOTE: This feature is disabled by default.

9.1 Usage

The Remote Escape Sequence feature utilizes an existing SPP profile link. To enable remote control, use the follow configuration command:

AT+AB Config RmtEscapeSequence = true

AT+AB Reset

A SPP connection must be established first. Once connected, user must send the Remote Escape Sequence "@#@$@%" to the remote device, which will respond locally with "AT-AB RemoteMode. Then, it will accept and respond (both locally and over the link) to all standard AT commands send remotely.

- The Remote Escape Sequence "@#@$@%" must be send in a single complete packet over the BT link.
- There should NOT be a CR or LF in this sequence.

10 Power Saving Mode

abSerial devices support various features, which allow low power operation over a range of scenarios. This section will discuss the Deep Sleep Mode, Sniff, and Auto Sniff features and how they may be effectively used.

NOTE: This feature is disabled by default.

10.1 Deep Sleep Mode

In abSerial, the basis for low power operation is Deep Sleep Mode, DSM. This feature temporarily halt's the chip's operation by stopping the main crystal and switching to the low power 32KHz oscillator instead. When enabled, DSM automatically enters this halt state whenever possible. Scheduled CPU activity, timers, remote link activity, and GPIO wakeup will automatically resume active mode operation.

10.1.1 UART Usage with Deep Sleep

The main UART is turned off while in Deep Sleep mode, and not commands or data may be sent. Typically, GPIO lines for Host Wakeup and Keep Awake are used to block DSM, in order to use the UART.

10.1.2 Deep Sleep Wakeup

abSerial supports a Deep Sleep Wakeup feature using a GPIO. When enabled, an active signal on the GPIO will temporarily prevent or block DSM. Normal DSM operation will resume when this signal is no longer asserted.

10.2 Sniff

abSerial supports sniff mode using an AT command, see the *abSerial Reference Guide* for details. When a connection is placed into sniff mode on a deep sleep enabled device, it will enter deep sleep during the inactive intervals between sniff polls. Since UART data cannot be received or transmitted when in deep sleep, communication will be blocked during the sniff intervals.

11 GPIO Usage

GPIO pins are supported on Amp'ed RF Bluetooth devices (see individual data sheets for GPIO details.) Some abSerial features require these pins when they are enabled; see table below. Also, AT commands can be used to control these pins.

11.1 abSerial Application GPIO PIN Assignments

This table gives a summary of the abSerial interface's assignment of certain GPIO pins. Only the GPIO pins directly used by the abSerial interface are considered in this table; for complete GPIO assignments see the applicable device Data Sheet.

GPIO	abSerial Usage
1	CPU/Deep Sleep activity, output
4	Link Connection, output

11.2 GPIO AT Commands

abSerial AT commands can configure, read, and write to any of the GPIO pins. If a pin is already in use by one of the above features, using an AT command to modify the pin will cause a conflict; this is not recommended. Refer to the *abSerial Reference Guide* for GPIO AT command details.