kcSerial 2.2 User Guide

21 May 2006

TABLE OF CONTENTS

1 Preface

The document describes an embedded application that provides a wirefree serial cable replacement service using the Bluetooth Serial Port Profile, kcSerial. Formally, this software was developed by Zeevo Inc. and was called Zerial.

1.1 Purpose

This document provides guidelines for users of host applications that use the kcSerial interface. Such host applications may execute on a personal computer or other device that will primarily pass a serial data stream to the kcSerial interface, which will then be transferred using either the Bluetooth Serial Port Profile (SPP) or Dial Up Network Profile (DUN).

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

This document will not explain how to construct an embedded application on the ZV4002, nor explain the implementation details of the kcSerial interface.

1.2 Definitions and Acronyms

The following acronyms are used in this document.

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

1.3 Feedback

We are constantly improving our product and would very much like to get your feedback. Please send your feedback in an email to support@kcwirefree.com.

For the latest updates and additional information please visit the KC Wirefree website at: www.kcwirefree.com

2 Overview

2.1 Modes of Operation

The software behavior of the kcSerial interface is similar to a Hayes-compatible modem. The application has two modes, a command mode and a bypass mode. In the command mode, the host can issue specially formatted text strings called commands. These command strings can be used for configuration or to manage a connection with a remote device. Note that the kcSerial interface does not support the standard Hayes AT command set. Instead, it has commands that leverage off the vendor-specific command form.

Once a connection is established, the application transitions to the bypass mode. In the bypass mode, bytes sent from the host will be sent over the Bluetooth link to the remote device. Any data received from the remote device will be delivered to the host.

All bytes received on the UART by the host are transferred to the remote device with the exception of the Escape sequence. While in the bypass mode, the kcSerial interface will search for this Escape sequence from the host. If this sequence is found, the application will go back to command mode. This allows commands to be issued again from the host, but the connection to the remote device will remain. Any data received on the Bluetooth link will be discarded while in command mode.

While in the command mode, the kcSerial interface will send responses back to the host for commands received. Responses from the kcSerial interface will also be sent on system reset. These responses are also referred to as Event strings in this document. Event strings are not sent to the host during bypass mode. However, it is possible to configure a disconnect notification to be sent during bypass mode. Please refer to the *kcSerial Reference Guide* for more details.

2.2 System Configuration

Hardware

To connect the host to the kcSerial interface, an RS-232 port is used with a NULL-modem cable. This cable will provide the connections illustrated below. The default port speed of the application is 115200 bps, with 8 data bits, no parity, and 1 stop bit. The supported serial port speeds are 9600 bps, 19200 bps, 38400 bps, 57600 bps, 115200 bps, 230400 bps, 460800 bps, and 921600 bps. Hardware flow control (RTS/CTS) is used by the kcSerial interface. If CTS is de-asserted, the application will stop its transmission within one byte. When the kcSerial interface de-asserts RTS, by default it can accept up to 22 more bytes (i.e., the host is expected to stop transmitting within 22 bytes).

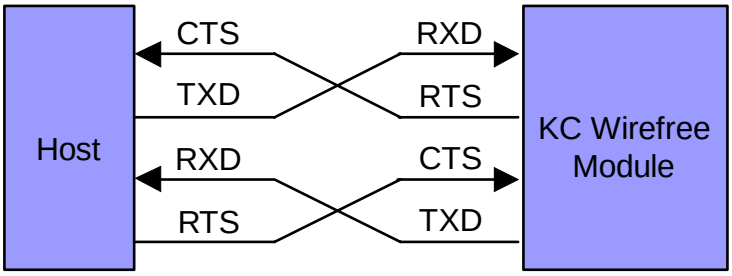


Figure 1. Physical Connection between Host Platform and Wirefree Serial Interface

Software

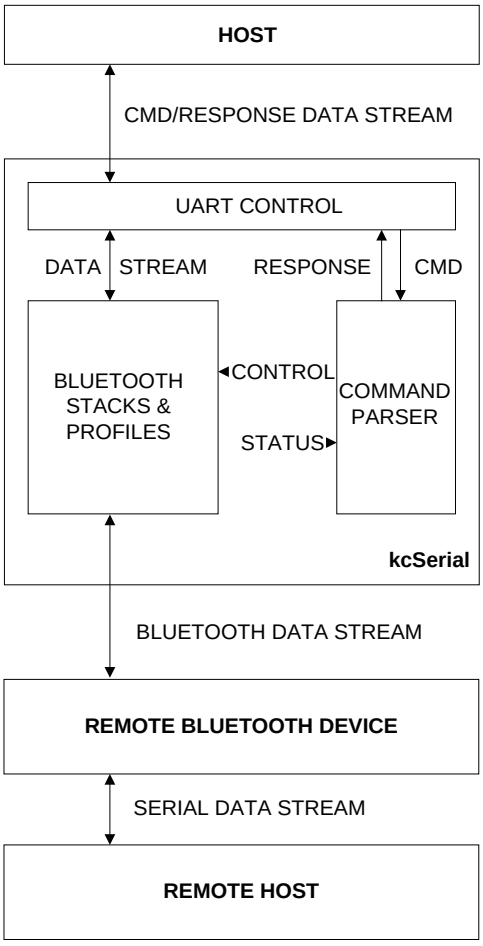
The kcSerial interface executes on the device connected physically to the local host. The remote device must at minimum be a device that supports the Serial Port Profile (SPP) or Dial Up Network Profile (DUN).

If deep sleep is enabled, RTS will be de-asserted (CTS on the local host side) while the Bluetooth radio is in deep sleep. In order for the application to accept data or commands again, the local host must pulse its RTS line. Within 35ms of the pulse, the firmware will accept commands again.

2.3 System Data Flow

The following diagram shows how the data stream into and out of the kcSerial 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.



2.4 Commands and Responses

A summary of all commands and responses can be found in the *kcSerial Reference Guide*.

3 Startup

Upon initialization (due to power up or system reset), the kcSerial 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 kcSerial interface is in command mode and the other lists the BD address of the local device.

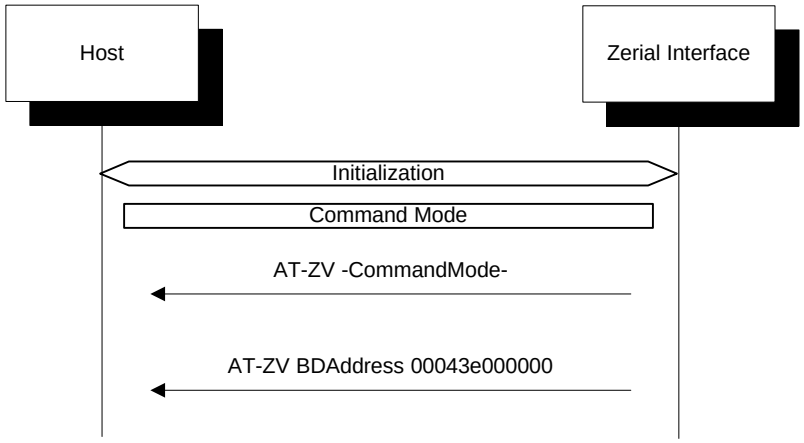


Figure 2. Event Strings Sent on Startup

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

4 Configuration

The configuration of the kcSerial interface allows the user to tailor the interface for a specific environment. This chapter addresses configuration through the use of AT commands; please refer to the *kcSerial Reference Guide* for a description of these commands.

Configurations may be applied following system startup of the kcSerial interface. The following subsections will describe common ways of using the configuration in different situations.

4.1 Master device configuration

If the kcSerial interface is being used as a master device, e.g., used to initiate connections, a common startup configuration may consist of setting the security mode.

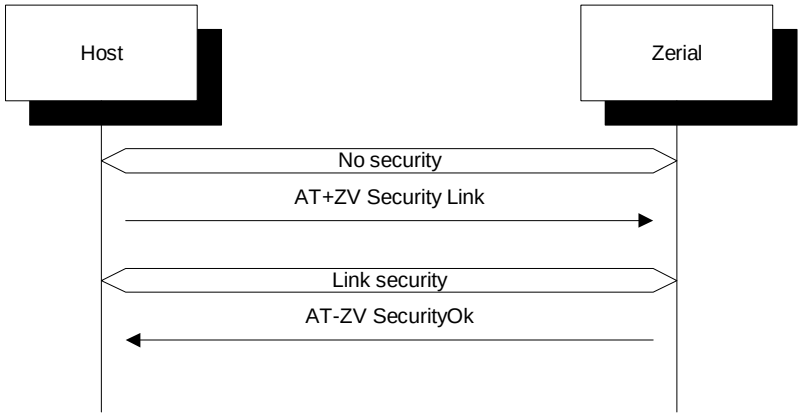


Figure 3. Command and Event Strings for Master Device Configuration

4.2 Slave device configuration

If the kcSerial interface is being used as a slave device, e.g., used to accept remote connections, a common startup configuration may consist of setting the local name and/or the security mode.

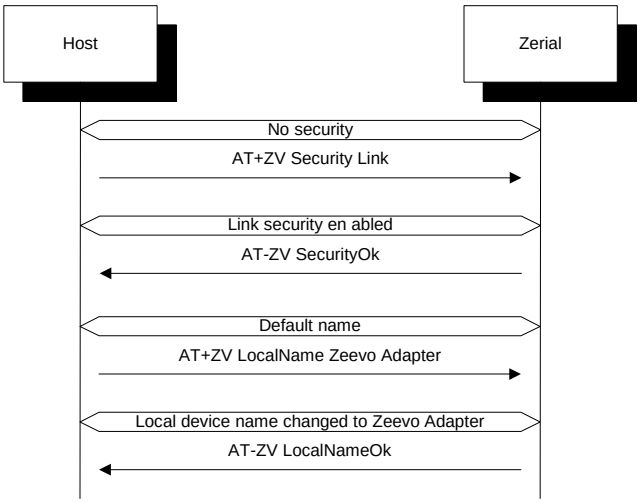


Figure 4. Command and Event Strings for Slave Device Configuration

4.3 kcSerial interface-to-Host Baud Rate

The host and kcSerial interface can communicate at a number of baud rates. It may be necessary for the host to temporarily change the baud rate from the baud rate used at power up (the default Dynamic Configuration is 115200 bps). The host can change the baud rate by changing the serial port speed while in command mode. The new port speed becomes effective once the response string has been transmitted to the host. This baud rate change remains in effect until the next system reset or power down. Alternatively, a new default baud rate may be configured, which will take effect after the next system reset. The configured baud rate does not directly affect the transfer of data over the Bluetooth connection.

The possible baud rates are: 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600 bps.

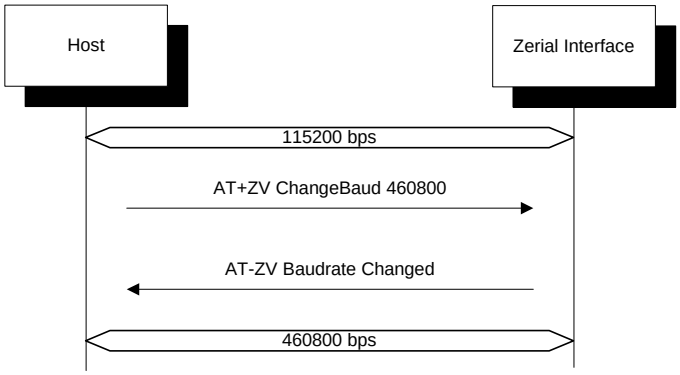


Figure 5. Command and Event Strings for Baud Rate Configuration

5 Connect with Remote Device

In order to create a connection with a remote device, the host issues a command string to the kcSerial 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.

5.1 Successful Connection

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

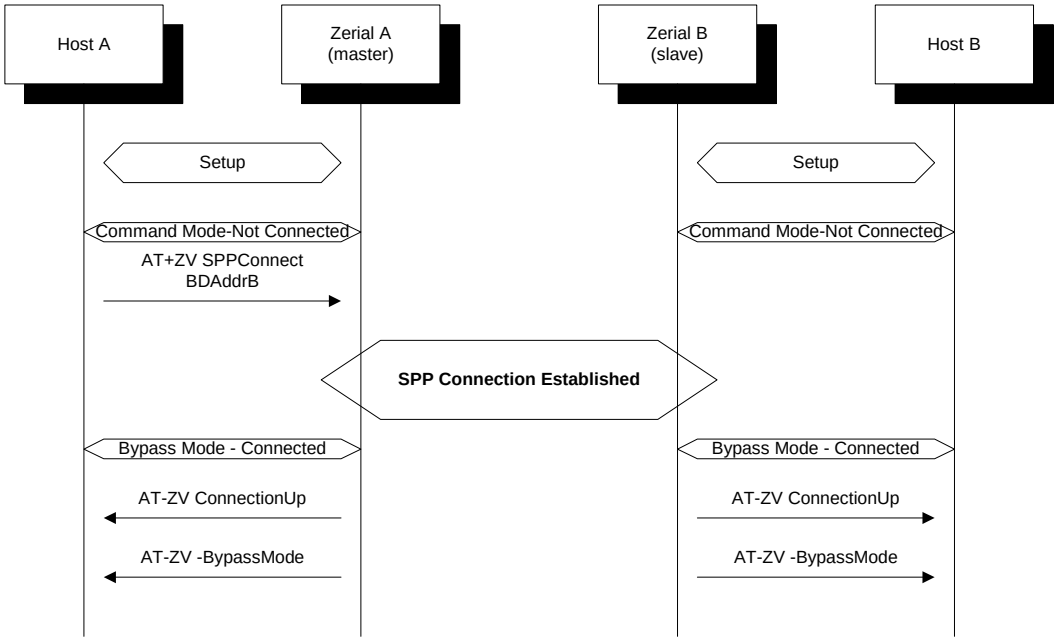


Figure 6. Command and Event Strings for Successful Connection

5.2 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.

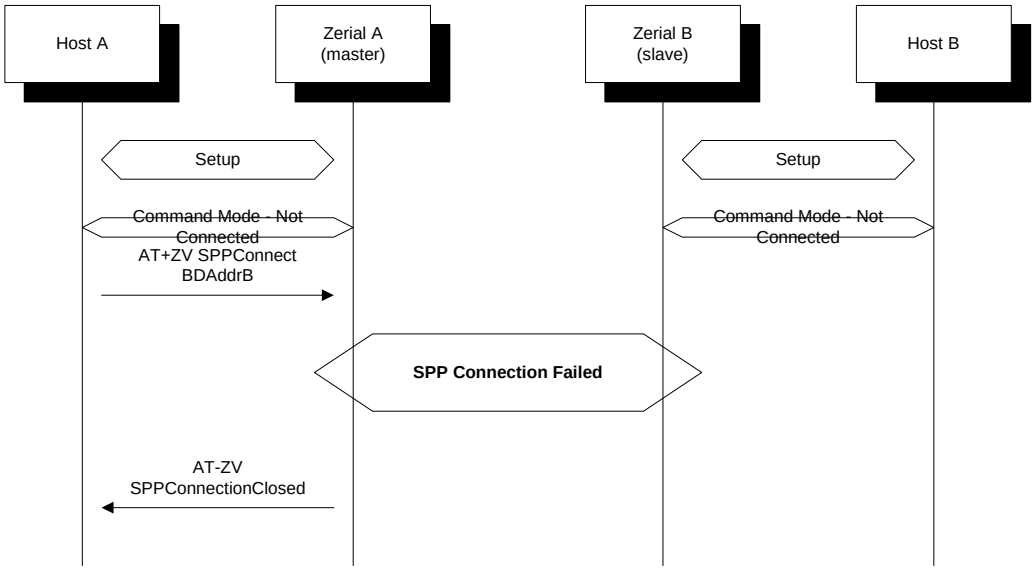


Figure 7. Command and Event Strings for Unsuccessful Connection

6 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 7.

Once the kcSerial 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.

6.1 Disconnect Notification Disabled

In the Dynamic Configuration, the kcSerial interface has a flag that enables notification on disconnect during bypass mode. By default, this is turned off. In that case, if the kcSerial interface is still in bypass mode while a remote device initiates a disconnect, the host has no notification.

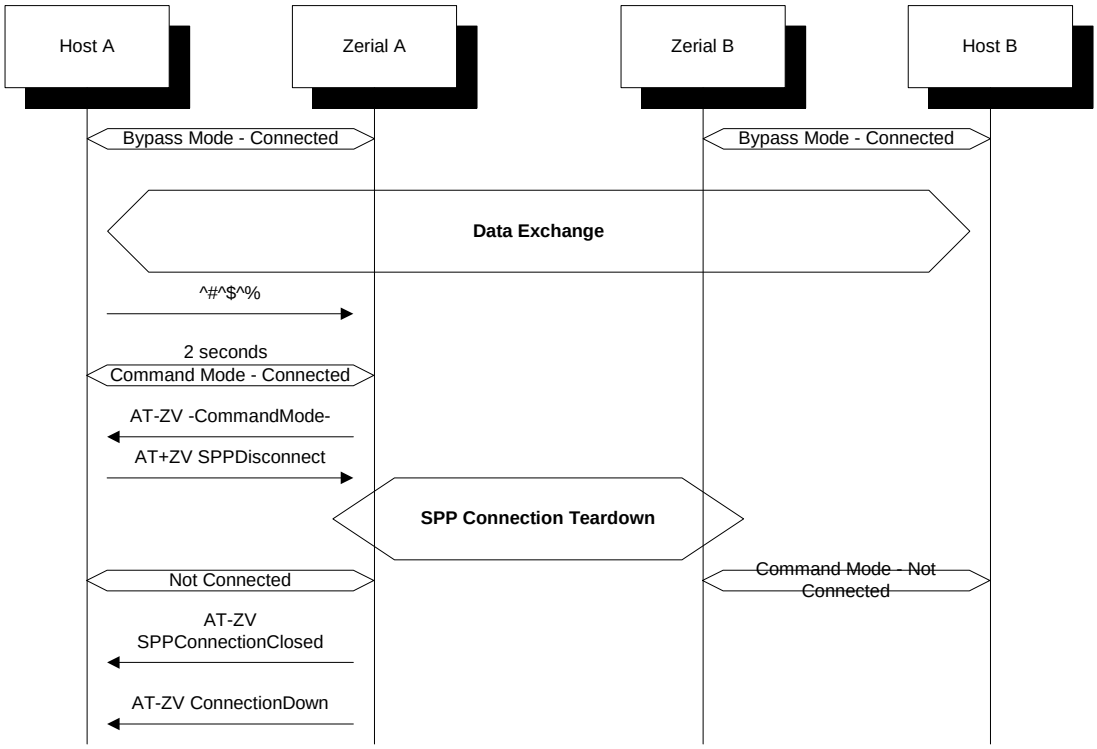


Figure 8. Escape, Command, and Event Strings for Notification Disabled

6.2 Disconnect Notification Enabled

In the Dynamic Configuration, the kcSerial interface has a flag that enables notification on disconnect during bypass mode. By default, this is turned off. If disconnect notification is enabled and the kcSerial interface is still in bypass mode while a remote device initiates a disconnect, the host receives notification.

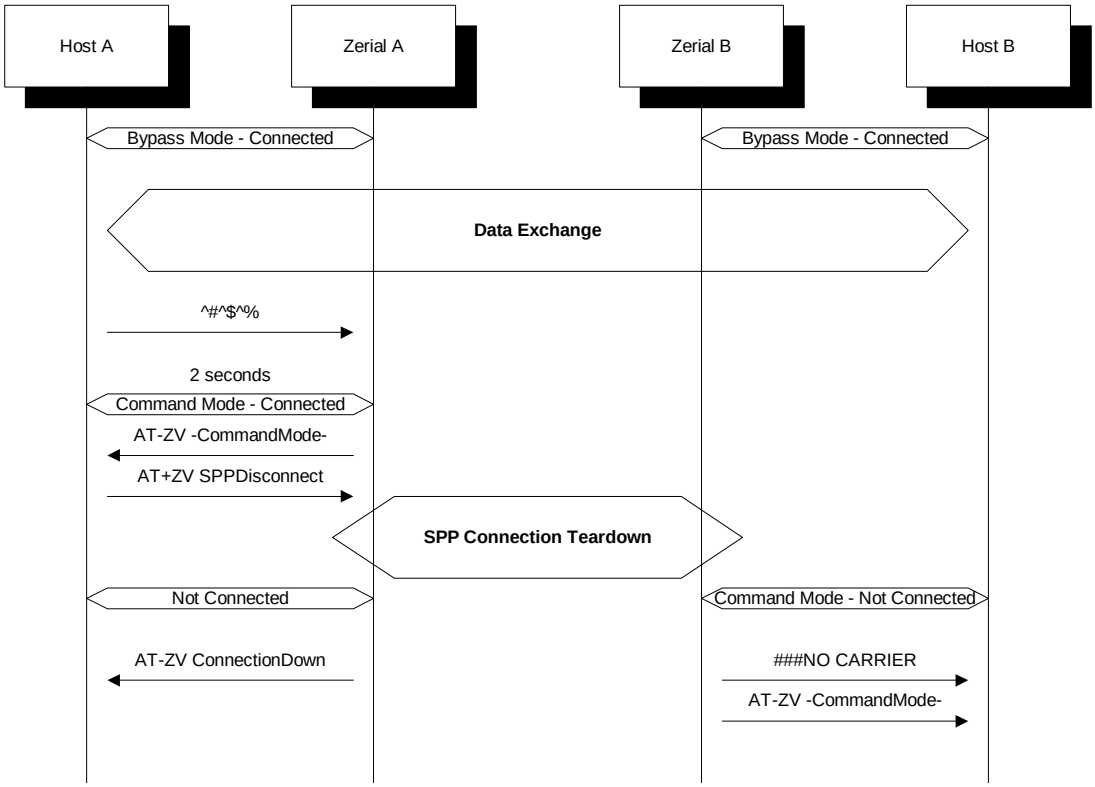


Figure 9. Escape, Command, and Event Strings for Notification Enabled

6.3 Both Devices in Command Mode

If both devices are already in command mode when the disconnect is initiated, the both hosts will receive notification.

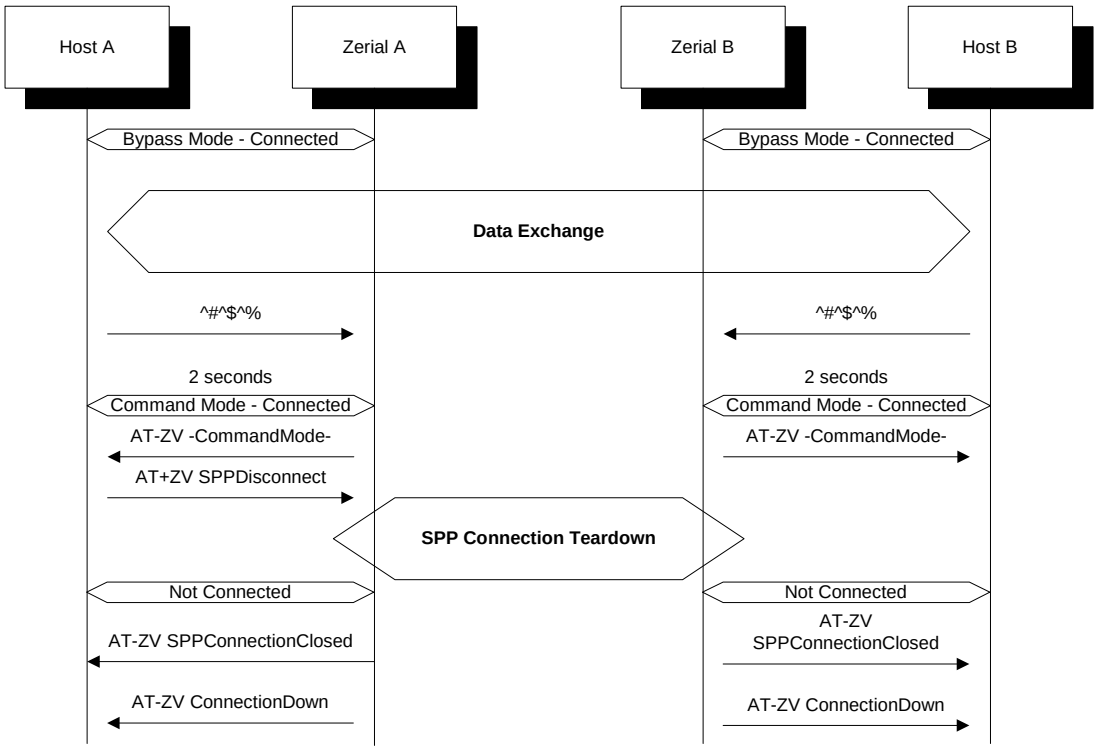


Figure 10. Escape, Command, and Event Strings for Command Mode

7 Escape from Bypass Mode

Once a connection has been established between host and remote device, the host can put the kcSerial 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 2 seconds 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 Wirefree Serial interface and not passed to the local host.

7.1 Connection Still Active

If the kcSerial interface is in bypass mode when the escape string is sent (i.e., the connection is still active), the host must wait 2 seconds before the application will respond in the command mode.

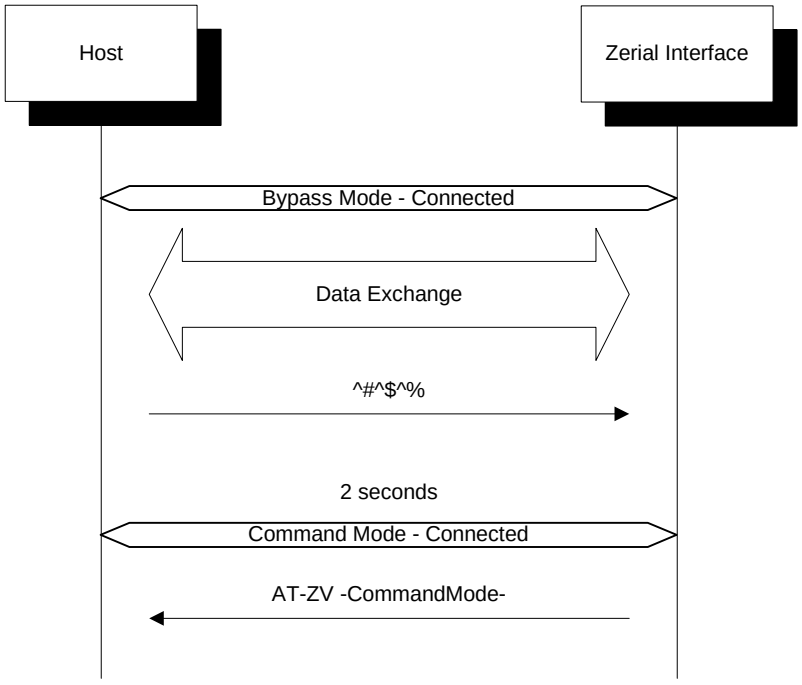


Figure 11. Escape Sequence and Event String when Connection Is Active

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

8 Device Discovery

A kcSerial 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, remote device name, and the available services. The kcSerial 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 *kcSerial 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. This can fail if:

1) The local device is unable to page the remote device and/or establish a connection.

2) A connection is established, but the remote device does not have a valid name registered by its host.

If this occurs, the system will return the message: "`Unknown`".

Once name discovery is complete, the device performs service discovery on the same remote device. The device: a) inquires for the remote device, b) pages the remote device and establishes a connection, c) performs service discovery and then d) disconnects. Service discovery is set to locate ALL services by default. The service discovery procedure can fail if:

3) The local device is unable to locate the remote device during inquiry.

4) The local device is unable to page and establish a connection.

5) The remote device does not have a registered SPP server.

If this occurs, the system will return the message: "`NoSvcs`".

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.

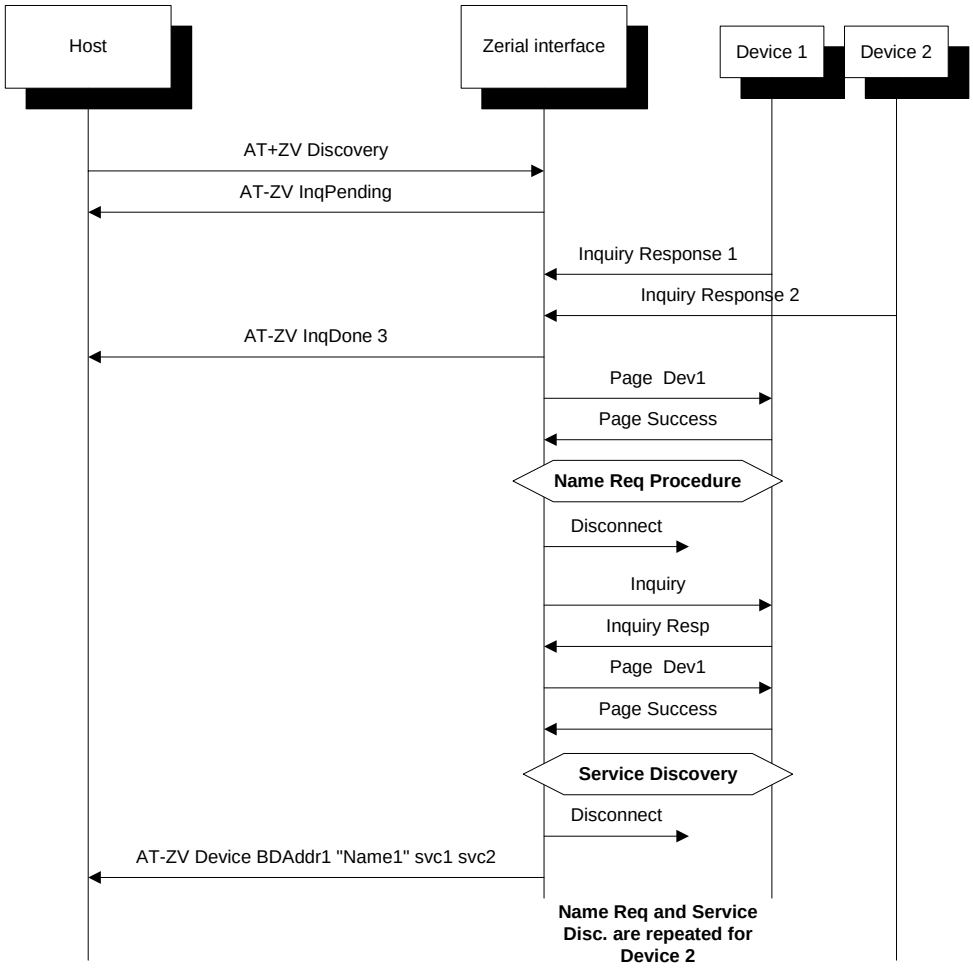


Figure 12. Commands and Events for Device Discovery

9 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.

Security and Bonding are enabled separately, but are both required to establish a secure device. When the security level is set to "`Link`" the device requires a LinkKey to identify and authenticate any remote devices. A remote device which has been paired or bonded to the local device, will have a valid LinkKey and a connection may be established. Without security, "`None`", no LinkKey is required between bonded devices. kcSerial allows up to four remote devices and their corresponding LinkKeys to be stored. In order to bond a new device and generate its LinkKey, the lower layer stack will ask the Host for a PIN, also called the PassKey.

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

a) Initiate authentication with the remote device (also known as LM_Pairing)

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

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

d) Store the new LinkKey for Device A.

In c), if one of the devices does not verify the PIN, bonding fails and the connection is terminated.

Upon successive connections to DeviceA, the kcSerial device will automatically use the stored LinkKey to authenticate the remote device and initiate encryption without notifying the kcSerial host. To the user or host application, pairing occurs automatically. However, if the application disconnects DeviceA and connects to DeviceB, it will follow steps a) through d) all over again if DeviceB is not bonded to the local device.

For further details on "`Bond`", "`EnableBond`" , "`Security`" commands, please refer to the *kcSerial Reference Guide*.

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

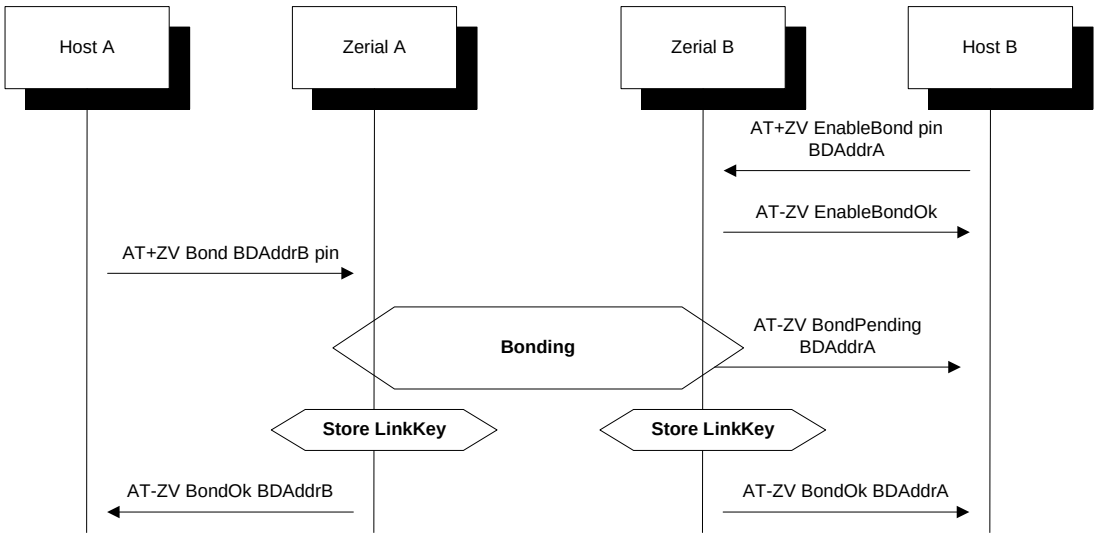


Figure 13. Command and Event Strings for Successful Bonding

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

For further details on "`Bond`", "`EnableBond`", and "`Security`" commands, please refer to the *kcSerial Reference Guide*.

10 Smart Cable

This feature provides a simple, user-friendly, cable replacement application, the Smart Cable. An initial configuration from either the dynamic configuration file or AT command interface is used to set up the Smart Cable parameters. Once this is done, no further user intervention is necessary during normal usage. This section details this feature's initial setup and usage.

10.1 Configuration Parameters

The following parameters are supported from both an AT command and the dynamic configuration file:

- The BD Address of the remote auto-connect device. A blank entry in the dynamic configuration, all 0's, will disable this feature if the AT command, Smart Cable setup, is not being used.
- Reconnect attempt interval - 100ms to 100 seconds in 100ms increments.
- Reconnect attempts – 0 to 999. A value of 1,000 signifies unlimited attempts.

10.2 Usage

- Automatically establishes a SPP link to its designated remote device.
- The designated device is paged and retried up to the retry attempt limit setting, if it is unable to connect initially.
- If a link is disconnected, the Smart Cable feature will automatically re-connect the link without user interaction.
- A wait interval is inserted between automatic page attempts.
- Only point-to-point connections are supported.
- An optional feature allows GPIO 7 to quick connect to the remote device as well as reset the current page retry attempt counter.

The Smart Cable setup AT command automatically updates its non-volatile memory parameters. These new settings are loaded after the next reset. The Delete Smart Cable AT command deletes these parameter settings in NVM and deactivates the Smart Cable feature for the remainder of the session. Upon reset, either the stored Smart Cable setup from the AT command, if in use, or the dynamic configuration setup is used. The AT command cable setup has priority over the dynamic configuration cable setup. If neither setup contains a remote BD address entry (all 0's), then the feature is disabled.

The GPIO 7 attempt reset feature resets the current connection attempt counter. If the attempt parameter is set to 0 attempts, a page is still sent (forced) 1 time.

11 Remote Command Mode

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

NOTE: This feature is disabled by default.

11.1 Usage

The Remote Command feature registers an additional SPP profile based service, called “Remote Cmd”. To remotely control a device, a SPP connection must be established to this service. Once connected, the remote device accepts and responds to all standard AT commands, over the Bluetooth command link, to a local host. The host of the remote device may still be used as well. A data link using the ZV-SPP service may be connected simultaneously and Bypass mode is supported while a command link is in use; RemoteCmdWithBypassState. Since both the data link and command link are active in Bypass mode (RemoteCmdWithBypassState state); there is no need to escape to Command mode or Remote Command Mode to send AT commands using the remote command link.

11.2 State Transition Matrix and Diagram

Two new application states have been added for the remote device, RemoteCmdNoBypassState and RemoteCmdWithBypassState.

	Command	**Bypass**	**Remote Cmd With Bypass**	**Remote Cmd No Bypass**
Escape Cmd	Command	Command	Remote Cmd No Bypass	Remote Cmd No Bypass
Bypass Cmd	Bypass	NA	NA	Remote Cmd With Bypass
Remote Cmd Link Down	NA	NA	Bypass	Command
Remote Cmd Link Up	Remote Cmd No Bypass	Remote Cmd with Bypass	NA	NA
SPP Link Up	Bypass	NA	NA	Remote Cmd with Bypass
SPP Link Down	NA	Command	Remote Cmd no Bypass	NA

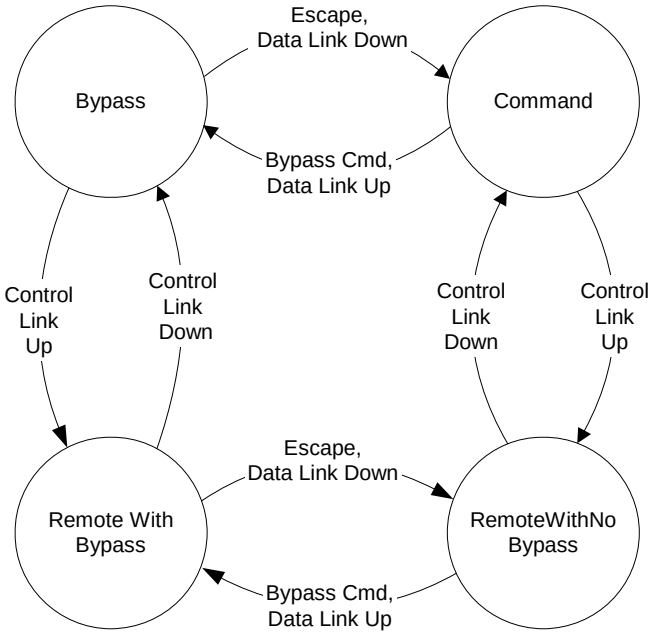


Figure 14. Remote Command State Diagram

12 Power Saving Mode

kcSerial 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.

12.1 Deep Sleep Mode

In kcSerial, 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, GPIO interrupts, and UART requests will automatically resume active mode operation.

UART Usage With Deep Sleep

When a UART is connected, the CTS line on the device's UART connector must not be asserted in order to allow DSM. The host device design must consider this when DSM is desired. In order to wake up from DSM, the host must pulse the device's CTS line and wait 10ms for the device to become active once again.

Deep Sleep Blocking

kcSerial supports a Deep Sleep Blocking feature using GPIO 5. When enabled, an active signal on GPIO 5 will temporarily prevent or block DSM. Normal DSM operation will resume when this signal is no longer asserted.

Streaming Serial With Deep Sleep

The Streaming Serial feature should not be used with DSM. When no hardware flow control is supported, the Rx UART on the kcSerial device cannot receive data or halt its transmission while in deep sleep. This scenario will lose all of the data sent to the kcSerial device when DSM is active.

12.2 Sniff

kcSerial supports sniff mode using an AT command, see the *kcSerial 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. This may not be acceptable for many applications, so an application controlled Auto Sniff mode is supported.

Auto Sniff Mode

This feature dynamically enables and disables sniff mode depending on a link's communication needs. Two dynamic configuration parameters control this feature: Sniff Interval and Inactivity Timeout. The Sniff Interval is the number of sniff poll interval slots that the sniff mode uses. The Inactivity Timeout is the number of seconds that the link will stay active after data is received or transmitted.

13 GPIO Usage

Zeevo Bluetooth devices support 16 GPIO pins (8 on the TC2001). Some kcSerial features require these pins when they are enabled; see table below. Also, AT commands can be used to control these pins.

13.1 kcSerial Application GPIO PIN Assignments

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

All unassigned GPIO pins are defaulted to inputs.

GPIO	kcSerial Usage	Default
1	CPU/Deep Sleep activity LED 2, output	Enabled
4	Base band activity debug LED 1, output	Enabled
5	DSM Blocking, input: High; DSM blocked, Low; DSM allowed	Disabled
6	Streaming Serial, input: High; RTS/CTS enabled, Low; Streaming Serial enabled	Enabled
7	Smart Cable connection reset, input: Active high	Disabled

Note: GPIO inputs, 0-7 are internally pulled low, 8-15 are internally pulled high.

13.2 GPIO AT Commands

kcSerial AT commands can configure, read, and write to any of the 16 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 *kcSerial Reference Guide* for GPIO AT command details.

14 Feature Compatibility Matrix

The following table details the compatibility of currently supported features:

	Remote Command*Beta*	Streaming Serial	Smart Cable*Beta*	Auto Sniff*Beta*	DUN
Remote Command*Beta*		✓	✓		
Streaming Serial	✓		✓	✓	✓
Smart Cable*Beta*	✓	✓		✓	
Auto Sniff*Beta*		✓	✓		
DUN		✓			

Figure 15. Feature Compatibility Matrix

Any combination of features that are compatible is allowed.

kcSerial 2.2 User Guide
21 May 2006

Main Office

2640 W Medtronic Way
Tempe, Arizona 85281

(602) 386-2640 Phone
(602) 386-2642 Fax

info@kcwirefree.com

Engineering Office

1722 Ringwood Ave
San Jose, California 95131

(408) 850-2828 Phone
(408) 850-2829 Fax

tech@kcwirefree.com

kcSerial 2.2 Reference Guide

21 May 2006

TABLE OF CONTENTS

1 Preface

The document describes an embedded application that provides a kcSerial cable replacement service using the Bluetooth Serial Port Profile. This software was originally developed by Zeevo, Inc. under the name Zerial.

1.1 Purpose

This document provides a detailed description of each command supported by the kcSerial interface. Each description explains parameters and the expected behaviors of each command and response.

Errors responses are also detailed in this document.

1.2 Definitions and Acronyms

Table 1. Definitions and Acronyms

Term	Description/Meaning
ASCII	American Standard Code for Information Interchange, a standard describing encoding of characters; the use in this document is strictly US 7-bit
BD	Bluetooth Device
DCD	Modem signal “data carrier detect”; indication from a modem that a connection has been made through, for example, a dialup connection
DTE	Data terminal entity, e.g., a computer
DTR	Modem signal “data terminal ready”; indication to a modem that the data terminal is ready for a connection
DUN	Dialup Networking (Profile)
GPIO	General Purpose Input-Output
LAN	Local Area Network
PIN	Personal Identification Number
SPP	Serial Port Profile
UART	Universal Asynchronous Receiver-Transmitter

1.3 Feedback

We are constantly improving our product and would very much like to get your feedback. Please send your feedback in an email to support@kcwirefree.com.

For the latest updates and additional information please visit the KC Wirefree website at: www.kcwirefree.com

1.4 Versions

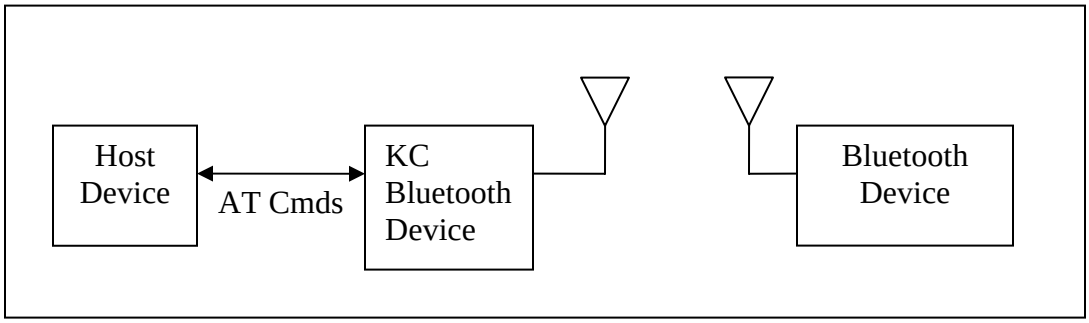
Later versions add additional features. kcSerial v2.2 corrects a bug, and contains the same features as v2.1.

2 Overview

This chapter gives a basic overview of the kcSerial interface. For further information, please refer to the *kcSerial User Guide*.

2.1 kcSerial Interface Overview

kcSerial is a cable replacement application that provides point-to-point communication between two Bluetooth devices. A serial port is used to communicate with a host device through an AT command interface as shown below.



kcSerial provides the following basic features:

- Point-to-point connection – kcSerial only supports a connection with one device at a time.
- Serial Port Profile – SPP is supported with kcSerial for both Client and Server application.
- Dial Up Networking – DUN profile support for Client applications (DUN Server is not currently supported).
- Command and Bypass modes – it is possible to switch between Command and Bypass (data transmit/receive) modes during an active connection
- Security – Bonding and data encryption provides a secure link between two devices.
- Multiple Device Bonding – special security keys can be exchanged with multiple devices to allow different devices to securely connect with kcSerial (although not simultaneously)
- Power conservation – use of Park, Sniff, and Hold features to minimize power consumption
- Variable Baud Rates – the serial port can be configured for the following baud rates: 9600, 19.2K, 38.4K, 57.6K, 115.2K (default), 230.4K, 460.8K, 921.6K

3 Commands

This chapter details the each of the kcSerial AT commands including brief descriptions of behavior, syntax of the command, context of the command, and types of responses.

Table Key:

✓ – command is supported in this release

U – command has been updated for this release, see release notes

X – command not supported in this release

Table 3: kcSerial Command Summary

Command	kcSerial v1.0	kcSerial v2.0	kcSerial v2.1 kcSerial v2.2
Bond	✓	✓	✓
Bypass	✓	✓	✓
ChangeBaud	✓	✓	✓
ChangeDefaultBaud	✓	✓	✓
DefaultLocalName	X	✓	✓
DeleteSmartCable	✓	✓	✓
DisableBond	✓	✓	✓
Discovery	✓	✓	✓
DunConnect	✓	✓	✓
DunDisconnect	✓	✓	✓
EnableBond	✓	✓	✓
EraseBondTable	✓	✓	✓
ExitPart	✓	✓	✓
ExitSniff	✓	✓	✓
GPIOConfig	✓	✓	✓
GPIORead	✓	✓	✓
GPIOWrite	✓	✓	✓
Hold	✓	✓	✓
HostEvent	✓	✓	✓
LocalName	✓	✓	✓
Park	✓	✓	✓
RemoteCommand	X	X	✓

Command	kcSerial v1.0	kcSerial v2.0	kcSerial v2.1 kcSerial v2.2
RemoteCmdDisconnect	✓	✓	✓
Reset	✓	✓	✓
Security	✓	✓	✓
SmartCableSetup	✓	✓	✓
Sniff	✓	✓	✓
SPPConnect	✓	✓	✓
SPPDisconnect	✓	✓	✓
StreamingSerial	X	X	✓
UpdateInquiryScan	✓	✓	✓
UpdatePageScan	✓	✓	✓
Version	✓	U	U

The following subsections describe each of these commands in detail, including a description of behavior, syntax (including possible parameter values), and types of responses.

Some responses will not be “immediate”. Where applicable, these will be noted and will include an approximate delay before response.

For commands with optional parameters, all possible forms will be listed under the syntax subsection.

Error responses are described in Section 4 Error Responses.

3.1 Bond

The `Bond` command is used to initiate bonding with a specified device. A personal identification number (PIN) is also required with this command.

Syntax

```
AT+ZV Bond [BD addr] [PIN]
```

Where `[BD addr]` is the BD Address of the remote device with which to bond and `[PIN]` is the PIN code to use (up to 16 characters).

Responses

If the request is successfully submitted, the response is:

```
AT-ZV BondPending [BD addr]
```

If the operation is successful, the response is:

```
AT-ZV BondOk
```

If the operation fails, the response is:

```
AT-ZV BondFail
```

3.2 Bypass

The `Bypass` command is used to return the kcSerial interface to the bypass mode, if a connection is still available. The possible use for this is to change a setting after a connection has been made (such as the UART baud rate). If the kcSerial interface does not have a connection, it will respond as if the connection is down.

Syntax

```
AT+ZV Bypass
```

Responses

If a connection is still available, the response is:

```
AT-ZV -BypassMode-
```

If there is currently no connection, the response is:

```
AT-ZV ConnectionDown
```

3.3 ChangeBaud

The host sends the `ChangeBaud` command in order to change the local UART speed to a new speed identified by the host. This setting will only remain in effect during the current session - until reset.

Syntax

```
AT+ZV ChangeBaud [rate]
```

where `[rate]` is the new baud rate:

- 9600
- 19,200
- 38,400
- 57,600
- 115,200
- 230,400
- 460,800
- 921,600

Responses

If the change is accepted, the response is:

```
AT-ZV Baudrate Changed
```

The actual change will not occur until the response has been completely transmitted.

3.4 ChangeDefaultBaud

The host sends the `ChangeDefaultBaud` command in order to change the default UART speed to a new speed identified by the host. This command is used to override the default baud rate from the Dynamic Configuration script so that the device does not require reprogramming to update this setting. The new baud rate is updated permanently until the device is either re-programmed or another `ChangeDefaultBaud` command is issued. The baud rate specified in the command will not take effect until the device is reset. To change the baud rate of the current session, use the `ChangeBaud` command.

Syntax

```
AT+ZV ChangeDefaultBaud [rate]
```

where `[rate]` is the new baud rate:

- 9,600
- 19,200
- 38,400
- 57,600
- 115,200
- 230,400
- 460,800
- 921,600

Responses

If the change is accepted, the response is:

```
AT-ZV Baudrate Changed
```

3.5 DefaultLocalName

The `DefaultLocalName` command is used to set the name of the device to the name that is reported during device discoveries. By default, the kcSerial interface uses “KCWirefreeDevice”. Changing the name using this command will permanently change the local name, unlike the `LocalName` command.

Syntax

```
AT+ZV DefaultLocalName [name]
```

Where `[name]` is a string for the new local name (up to 50 characters). The space character is allowed; the name is assumed to be all text up to the end of the command.

Responses

If the operation is successful, the response is:

```
AT-ZV LocalNameOk
```

3.6 DeleteSmartCable

The `DeleteSmartCable` command removes the current Smart Cable settings that were entered using the SmartCableSetup command, but not the setting from the dynamic configuration. The Smart Cable will then be deactivated for the remainder of this session. Upon reset, if a dynamic configuration for a Smart Cable exists, it will be activated. If there is no dynamic configuration Smart Cable setup, then this feature will remain deactivated.

Syntax

```
AT+ZV DeleteSmartCable
```

Responses

If the operation is successful, the response is:

```
AT-ZV DeleteSmartCableDone
```

3.7 DisableBond

The `DisableBond` command is used to disallow new bonding with a device.

This command cannot be used while a connection is active.

Syntax

```
AT+ZV DisableBond
```

Responses

If the operation is successful, the response is:

```
AT-ZV BondDisabled
```

3.8 Discovery

The `Discovery` command is used to initiate a device discovery. The command will return the number of responses of nearby devices and then the individual responses with BD address, name of device, and service names - optional. The number of devices returned is limited to 10; and the number of services per device is limited to 8. Inquiry is performed with an interval of 10.24 seconds. The devices are reported in the same order as the original inquiry results.

Syntax

```
AT+ZV Discovery

AT+ZV Discovery [CoD]

AT+ZV Discovery [CoD] [profile] [include service
enable/disable]
```

where devices are filtered on the `[CoD]` class:

- All (default)
- Misc
- Computer

- Phone
- LAN
- Peripheral
- Imaging
- Unclass

where the service name is requested for the profile named `[profile]:`

- All (default)
- SPP
- DUN
- LAP
- FAX

where `[include service enable/disable]: true` returns both the remote services and names, and `false` skips the remote service discovery and only returns the remote names – this completes the discovery process faster.

Responses

When the discovery command has been accepted, the response is:

```
AT-ZV InqPending
```

Once the initial inquiry is complete and discovery has been started, the response is:

```
AT-ZV DiscoveryPending [num]
```

where `[num]` is the number of devices found, in decimal (up to 10 will be reported).

For each name or service name request that is successful, the response uses the returned names in the following format.

```
AT-ZV Device [BD addr] [name] [service name] [...]
```

where `[BD addr]` is in hexadecimal with the most significant byte first. `[name]` is a string in double quotes " ". `[service name]` is a string without quotes and is reported for each service reported.

For each unsuccessful name request, the corresponding name is replaced by `" Unknown"` . The name request may not be successful if unable to make a connection for the request.

```
AT-ZV Device [BD addr] " Unknown"  [service name] [...]
```

For each service request that does not return services, one name is returned as "`NoSvcs"`. A request may not return services if unable to connect to the device for the request or if the device does not contain the requested service.

```
AT-ZV Device [BD addr] [name] NoSvcs
```

For each unsuccessful service request, one name is returned as "`SvFail"`. This will only occur due to an internal error.

```
AT-ZV Device [BD addr] [name] SvFail
```

If the memory allocated for the service discovery buffers is insufficient, then then result may return an "Inval" response. If this occurs, please contact support@kcwirefree.com for assistance.

```
AT-ZV Device [BD addr] [name] [service name 1] [Inval] [Inval]
```

3.9 DUNConnect

The DUNConnect command is used to initiate a connection with the specified device. The remote BD address must be specified. The remote Service is optional. If not specified, the first registered DUN service will be used by default.

Syntax

```
AT+ZV DUNConnect [BD Addr] [Service]
```

Where [BD Addr] is the remote devices BD Address to page. [Service] is the specific service on the remote device; optional.

Responses

If the connection is successful, the response is:

```
AT-ZV ConnectionUp
AT-ZV -BypassMode-
```

If the connection cannot be completed, the response is:

```
AT-ZV DUNConnectionClosed
```

3.10 DUNDisconnect

The DUNDisconnect command is used to terminate a connection with the remote device.

Syntax

```
AT+ZV DUNDisconnect
```

Responses

If the connection is successful, the response is:

```
AT-ZV DUNConnectionClosed
```

3.11 EnableBond

The EnableBond command is used to enable bonding with another device. The BD Address, PIN and timeout parameters are optional.

When no BD Address is specified, requests from all BD Addresses are allowed.

If a BD Address is specified, bonding requests from devices with BD Addresses other than the one specified will fail and the existing link key will be deleted for that device.

Optionally, a PIN code may be entered with this command. If no PIN code is specified, the default PIN code will be used. The default PIN code is either the last 4 digits of the device's BD address or the dynamically configured PIN code, depending on the default PIN selection in the dynamic configuration file.

Also, a timeout value, in seconds, may be entered after the PIN code. Bonding will be disabled automatically after the requested timeout. If no timeout is specified, bonding is enabled until reset or until the `DisableBond` command is used.

If this command is issued multiple times, only the last PIN and BD address are saved. Also, if this command is issued before the first timeout occurs, the subsequent command will extend the timeout. The timeout is always set to the specified time beyond the last received `EnableBond`.

Syntax

```
AT+ZV EnableBond
AT+ZV EnableBond [BD addr]
AT+ZV EnableBond [BD addr] [PIN]
AT+ZV EnableBond [BD addr] [PIN] [timeout]
```

Where `[BD addr]` is the BD Address of the remote device with which to bond, `[PIN]` is the PIN code to use (up to 16 characters), and `[timeout]` is the duration of the timeout in seconds (1 to 14,400, in decimal).

Responses

If the operation is successful, the response is:

```
AT-ZV BondEnabled
```

If bonding has been initiated by a remote device, the notification is:

```
AT-ZV BondPending [BD addr]
```

where `[BD addr]` is the BD address of the remote device that initiated the bonding.

If bonding has occurred, the notification is:

```
AT-ZV BondOk [BD addr]
```

where `[BD addr]` is the BD address of the remote device with successful bonding.

If bonding was initiated by a remote device but failed, the notification is

```
AT-ZV BondFail
```

When the time limit for bonding has expired, the notification is

```
AT-ZV BondDisabled
```

3.12 EraseBondTable

The `EraseBondTable` command is used to erase all of the bonded device entries. Single devices cannot be erased with this command

Syntax

```
AT+ZV EraseBondTable
```

Responses

If the operation is successful, the response is:

```
AT-ZV BondTableErased
```

3.13 ExitPark

The `ExitPark` command is used to switch a device from park mode to active mode.

Syntax

```
AT+ZV ExitPark [BD address]
```

Where `[BD address]` is the BD address of the device to be switched to active mode.

Responses

If the operation is successful, the response is:

```
AT-ZV ActiveMode
```

3.14 ExitSniff

The `ExitSniff` command is used to switch a device from sniff mode to active mode.

Syntax

```
AT+ZV ExitSniff [BD address]
```

Where `[BD address]` is the BD address of the device to be switched to active mode.

Responses

If the operation is successful, the response is:

```
AT-ZV ActiveMode
```

3.15 GPIOConfig

The `GPIOConfig` command is used to configure a GPIO pin to input or output.

Syntax

```
AT+ZV GPIOConfig [GPIO Pin] [Configuration]
```

Where `[GPIO Pin]` is the Pin number, 0 – 15, of the desired GPIO to configure. `[Configuration]` is “i” or “I” for input and “o” or “O” for output.

Responses

If the operation is successful, the response is:

```
AT-ZV GPIOConfigDone
```

3.16 GPIORead

The `GPIORead` command is used to read a GPIO pin. A GPIO may be read while configured as either an input or output.

Syntax

```
AT+ZV GPIORead [GPIO Pin]
```

Where `[GPIO Pin]` is the Pin number, 0 – 15, of the desired GPIO to read.

Responses

If the operation is successful, the response is:

```
AT-ZV GPIOReadDone [result]
```

Where `[result]` is either a 1 to indicate high, or 0 to indicate low.

3.17 GPIOWrite

The `GPIOWrite` command is used to set a GPIO pin to high or low. A GPIO may only be set when configured as an output.

Syntax

```
AT+ZV GPIOWrite [GPIO Pin] [Setting]
```

Where `[GPIO Pin]` is the Pin number, 0 – 15, of the desired GPIO to read. `[Setting]` is a 1 to set a pin to high and a 0 to set a pin to low.

Responses

If the operation is successful, the response is:

```
AT-ZV GPIOWriteDone
```

3.18 Hold

The `Hold` command is used to switch a device from active mode to hold mode.

Syntax

```
AT+ZV Hold [BD address] [Hold Duration]
```

Where `[BD address]` is the BD address of the device to be switched to active mode. `[Hold Duration]` is given in slots from 4-10,000.

Responses

If the operation is successful, the response is:

```
AT-ZV HoldMode
```

3.19 HostEvent

The `HostEvent` command is used to enable/disable the host notification strings. This will override the default setting in the dynamic configuration only for the current session; until reset.

Syntax

```
AT+ZV HostEvent [Enable/Disable]
```

Where `[Enable/Disable]` is an “e” or “E” character to enable this parameter and a “d” or “D” character to disable it.

Responses

If the feature is successfully enabled, the response is:

```
AT-ZV HostEvent Enabled
```

If the feature is successfully disabled there is no response because the events have been disabled.

3.20 LocalName

The `LocalName` command is used to set the name of the device to the name that is reported during device discoveries. By default, the kcSerial interface uses “KCWirefreeDevice”. Changing the name using this command does not permanently change the local name.

Syntax

```
AT+ZV LocalName [name]
```

Where `[name]` is a string for the new local name (up to 50 characters). The space character is allowed; the name is assumed to be all text up to the end of the command.

Responses

If the operation is successful, the response is:

```
AT-ZV LocalNameOk
```

3.21 Park

The `Park` command is used to switch a device from active mode to park mode.

Syntax

```
AT+ZV Park [BD address] [Beacon Period]
```

Where `[BD address]` is the BD address of the device to be switched to active mode. `[Beacon Period]` is given in slots from 200-10,000.

Responses

If the operation is successful, the response is:

```
AT-ZV ParkMode
```

3.22 RemoteCommand

The `RemoteCommand` command is used to enable/disable the remote command mode. This setting is stored in persistent memory, and will be retained after each reset. Additionally, the new setting will take effect upon the next device reset.

Syntax

```
AT+ZV RemoteCommand [Enable/Disable]
```

Where `[Enable/Disable]` is an “e” or “E” character to enable this parameter and a “d” or “D” character to disable it.

Responses

If the feature is successfully applied, the response is:

```
AT-ZV RemoteCommand [Enabled/Disabled]
```

3.23 RemoteCmdDisconnect

The `RemoteCmdDisconnect` command is used to disconnect a remote command connection. This command only applies to the server side of the link; the remote device. The client or local device, if using the kcSerial interface, should use a SPPDisconnect command since it is not in Remote Command mode.

Syntax

```
AT+ZV RemoteCmdDisconnect
```

Responses

If the operation is successful, the response is:

```
AT-ZV RemoteCmdModeClosed
```

3.24 Reset

The `Reset` command is used to reset the kcSerial interface. This is provided in the event that a host application wants to perform a software reset for error recovery. There is a response prior to reset to verify the command was received by the kcSerial interface.

Syntax

```
AT+ZV Reset
```

Responses

If the operation is successful, the response is:

```
AT-ZV ResetPending
```

3.25 Security

The `Security` command is used to set the security level of the device in use. By default, security level none is used.

- Variable pin type (the pincode request event will always be received by the application from the stack), and
- 128-bit unit key.

Service level security, level 2, is not currently supported. The security setting is not preserved in non-volatile memory.

Syntax

```
AT+ZV Security [level]
```

where `[level]` is the type of security to use:

- None
- Link (default)

Responses

If the operation is successful, the response is:

```
AT-ZV SecurityOk
```

3.26 SmartCableSetup

The `SmartCableSetup` command is used enable and configure a Smart Cable device. A device's BD Address is specified with which to automatically establish a connection; replacing the need for AT connection commands. This command will override the dynamic configuration of a Smart Cable device until the `DeleteSmartCable` command is issued; it is saved in non-volatile memory.

Syntax

```
AT+ZV SmartCableSetup [BD address] [Attempts] [Interval]
```

Where `[BD address]` is the BD address of the remote device to page and attempt to connect. `[Attempts]` 0 – 999 is the number of pages the will be attempted to the specified device until a connection is successful. A value of 0 will not automatically page the remote device, however, GPIO 7 may be asserted to manually send a page. A value of 1000 will perform unlimited pages until connected. be switched to active mode. `[Sniff Interval]` is given in slots from 66-10,000. [Interval] 1-1000 is the number of 100ms intervals (0.1sec to 100 sec) between page attempts. This interval is in addition to the amount of time required by the page attempt itself.

Responses

If the operation is successful, the response is:

```
AT-ZV SmartCableConfigDone
```

3.27 Sniff

The `Sniff` command is used to switch a device from active mode to sniff mode.

Syntax

```
AT+ZV Sniff [BD address] [Sniff Interval]
```

Where `[BD address]` is the BD address of the device to be switched to active mode. `[Sniff Interval]` is given in slots from 66-10,000.

Responses

If the operation is successful, the response is:

```
AT-ZV SniffMode
```

3.28 SPPConnect

The `SPPConnect` command is used to initiate a connection with the specified device. The remote BD address must be specified. The remote Service is optional. If not specified, the first registered SPP service will be used by default.

Syntax

```
AT+ZV SPPConnect [BD Addr] [Service]
```

Where `[BD Addr]` is the remote devices BD Address to page`. [Service]` is the specific service on the remote device; optional.

Responses

If the connection is successful, the response is:

```
AT-ZV ConnectionUp
AT-ZV -BypassMode-
```

If the connection cannot be completed, the response is:

```
AT-ZV SPPConnectionClosed
```

3.29 SPPDisconnect

The `SPPDisconnect` command is used to terminate a connection with the remote device.

Syntax

```
AT+ZV SPPDisconnect
```

Responses

If the connection is successful, the response is

```
AT-ZV SPPConnectionClosed
```

3.30 StreamingSerial

Syntax

```
AT+ZV StreamingSerial [Enable/Disable]
```

Where [Enable/Disable] is an “e” or “E” character to enable this parameter and a “d” or “D” character to disable it.

Query

An alternative syntax may be used to query the current StreamingSerial feature status. This syntax is not supported by other commands.

```
AT+ZV StreamingSerial
```

Responses

If the feature is successfully applied or queried, the response is:

```
AT-ZV StreamingSerial [Enabled/Disabled]
```

3.31 UpdateInquiryScan

The UpdateInquiryScan command is used to modify the Inquiry scan parameters: mode, duration, and interval.

Syntax

```
AT+ZV UpdateInquiryScan [mode] [duration] [interval]
```

where[mode] is the discoverable mode:

- 0: non-discoverable
- 1: limited discoverability – NOT SUPPORTED
- 2: discoverable

[duration]

is the scan length in slots; 18 to 4096. The default duration is 18 slots.

[interval]

is the period between scans in slots; 18 to 4096. The default interval is 2048 slots

Responses

If the command is successful, the response is:

```
AT-ZV InquiryScanUpdateDone
```

3.32 UpdatePageScan

The UpdateInquiryScan command is used to modify the Page scan parameters: mode, duration, and interval.

Syntax

```
AT+ZV UpdatePageScan [mode] [duration] [interval]
```

Where [mode] is the connectable mode:

- 0: non-connectable
- 1: connectable

`[duration]` is the scan length in slots; 18 to 4096. The default duration is 18 slots.

`[interval]` is the period between scans in slots; 18 to 4096. The default interval is 2048 slots

Responses

If the command is successful, the response is:

```
AT-ZV PageScanUpdateDone
```

3.33 Version

The `Version` command is used to return the current version of the kcSerial interface.

Syntax

```
AT+ZV Version
```

Responses

If the operation is successful, the response is:

```
AT-ZV kcSerialVer [x.y]
```

where `[x.y]` is the current version of the kcSerial Interface.

4 Error Responses

This chapter details the error responses that occur under specific circumstances.

There are seven error responses that can occur beyond error responses specific to a particular command (e.g., `Discovery`). They are:

`ErrConnect`	`ErrInvalidParam`
`ErrExecution`	`ErrNumParam`
`ErrFormat`	`ErrUnknownCmd`
`ErrInProgress`	

The following subsections detail the different error responses. These error responses replace the original error response

`AT-ZV ErrorCommand`

4.1 ErrConnect

The `ErrConnect` error response will be sent if kcSerial has a valid connection established and the command cannot be executed while connected (even in the command mode). Examples of commands that produce this error response are given in the following table.

Table 2. Example Commands that Generate <ErrConnect>

Example	Reason
`AT+ZV Security None`	Changing security level while a connection is up.
`AT+ZV Discovery`	Performing a device discovery while a connection is up.
`AT+ZV SPPConnect 00043e000000`	Establishing a connection to a device while a connection is exists with another.

4.2 ErrExecution

The `ErrExecution` error response will be sent if the command cannot complete for any reason.

Examples of commands that produce this error response are given in the following table.

Table 3. Example Commands that Generate <ErrExecution>

Example	Reason
`AT+ZV Security None`	Execution of command with correct syntax failed.
`AT+ZV Discovery`	Execution of command with correct syntax failed.

4.3 ErrFormat

The `ErrFormat` error response will be sent if kcSerial receives a command (text terminated by a carriage return or line feed) that does not match the expected format of starting with "AT+ZV".

Examples of commands that produce this error response are given in the following table.

Table 4. Example Commands that Generate <ErrFormat>

Example	Reason
`AT-ZV Discovery`	A valid command nam that does not start with the right prefix.
`SPPConnect 00043e000000`	A command does not start with AT+ZV
`abcdef`	A command does not start with AT+ZV

4.4 ErrInvalidParam

The `ErrInvalidParam` error response will be sent if the parameters for the requested command are not correct. The parameter(s) will be echoed back to the user starting from the parameter that was rejected.

Examples of commands that produce this error response are given in the following table.

Table 5. Example Commands that Generate <ErrInvalidParam>

Example	Response	Reason
`AT+ZV SPPConnect 8136`	`AT-ZV ErrInvalidParam 8136`	Numeric parameter not specified with required number of digits (BD address must always be 12 hex characters).
`AT+ZV SPPConnect 00043e008136 GOEP`	`AT-ZV ErrInvalidParam goep`	Unrecognized (or unsupported) symbolic parameter used.
`AT+ZV ChangeBaud 1600`	`AT-ZV ErrInvalidParam 1600`	Numeric parameter is out of range (specified baud rate is not supported by command).
`AT+ZV EnableBond 00043e000000 12345678901234567`	`AT-ZV ErrInvalidParam 12345678901234567`	String parameter (PIN)has too many characters.

4.5 ErrNumParam

The `ErrNumParam` error response will be sent if there are too few parameters for the requested command. A command sent with too many parameters does not generate an error; instead, the extra parameters are ignored.

Note: some commands will accept a variable number of parameters.

Examples of commands that produce this error response are given in the following table.

Table 6. Example Commands that Generate <ErrNumParam>

Example	Reason
`AT+ZV Security`	No parameters were specified.
`AT+ZV Sniff`	The minimum number of parameters was not specified.

4.6 ErrUnknownCmd

The `ErrUnknownCmd` error response will be sent if the requested command is not recognized. The unrecognized command will be echoed back to the host. Any parameters given will be ignored.

```
AT-ZV ErrUnknownCmd [unrecognized command]
```

An accepted command always starts with the command identifier:

```
AT+ZV
```

4.7 ErrInProgress

`ErrInProgress` is sent in response to `Discovery` command being issued when the previous one is still in progress.

```
AT+ZV Discovery [unrecognized command]
```

Examples of commands that produce this error response are given in the following table:

Table 7. Example Commands that Generate <ErrInProgress>

Example	Reason
`AT+ZV Discovery`	Trying to do discovery when the previous one has not completed.

4.8 Commands and Associated Errors

The table below summarizes which of the commands produce specific error responses. If a command can produce an error response, the column will be marked with an X.

Table 8. Possible ErrConnect Error Responses

Example	Err InvalidParam	Err NumParam	Err Execute	Err Connect	Err InProgress
Bond	X	X	X		
Bypass					
ChangeBaud	X	X			
ChangeDefaultBaud	X	X			
DefaultLocalName		X	X		
DeleteSmartCable					
DisableBond					
Discovery	X	X	X		X
DUNConnect	X	X		X	
DUNDisconnect					
EnableBond	X	X	X		
EraseBondTable					
ExitPark	X		X		
ExitSniff	X		X		
GPIOConfig	X	X			
GPIORead	X				
GPIOWrite	X	X			
Hold	X	X	X		
LocalName		X	X		
Park	X	X	X		
RemoteCmdDisconnect					
Reset					
Security	X	X	X		
SmartCableSetup	X	X			
Sniff	X	X	X		
SPPConnect	X	X		X	
SPPDisconnect					
UpdateInquiryScan	X	X	X		
UpdatePageScan	X	X	X		
Version					

5 Other Responses

The following subsections describe the 4 types of responses that occur under specific circumstances, not necessarily as a result of a specific command. They are:

- Reset
- Escape Sequence
- Controlled Disconnect
- Unexpected Disconnect

5.1 Reset

Upon either hardware reset or software reset (such as the `Reset` command), kcSerial will respond as follows after the reset is complete:

```
AT-ZV -CommandMode-
AT-ZV BDAddress [BD addr]
```

Because the BD address of the local device is reported during this response, the response is different than a response to the Escape Sequence.

5.2 Escape Sequence

If the Escape sequence `^#^$^%` is received and no connection is active, kcSerial will immediately respond with:

```
AT-ZV -CommandMode-
```

When the Escape Sequence is received while a connection is still active and there is no data for 2 seconds, kcSerial will respond (after those 2 seconds of no data) with the same string.

kcSerial will now be in command mode.

5.3 Controlled Disconnect

If the local host initiates a disconnect, it must first put the kcSerial interface into command mode (see the section on Escape Sequence directly above). After a successful disconnect command, the following response is made:

```
AT-ZV ConnectionDown
```

5.4 Unexpected Disconnect

Bluetooth connections may be unexpectedly dropped (e.g., in changing RF conditions). Although it is generally assumed that a disconnect will be “negotiated” on the application level, the remote device may initiate a disconnect. When that happens, the disconnect may be unexpected. This section applies to both the general case and the unexpected disconnect.

It is useful for the local host to be notified that a connection has been terminated when it isn't controlling the termination. An unexpected disconnect is essentially defined as a disconnect that occurs while in bypass mode. If this happens, kcSerial will respond with:

```
###NO CARRIER
AT-ZV -CommandMode-
```

It is the responsibility of the host to prevent this string from appearing in the data stream during normal operation.

If a remote disconnect occurs during command mode, this notification string is also sent. It will not be sent, however, if an initial setup cannot be established or if the disconnect is requested by the local device.

Hardware handshaking is not used to indicate a disconnection in this implementation. Modems can use DCD (data carrier detect) to notify the DTE (data terminal entity, e.g., computer) that a connection is either available or unavailable.

kcSerial 2.2 Reference Guide
21 May 2006

Main Office

2640 W Medtronic Way
Tempe, Arizona 85281

(602) 386-2640 Phone
(602) 386-2642 Fax

info@kcwirefree.com

Engineering Office

1722 Ringwood Ave
San Jose, California 95131

(408) 850-2828 Phone
(408) 850-2829 Fax

tech@kcwirefree.com