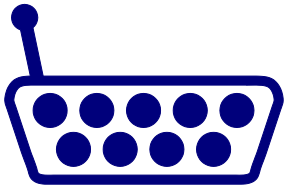


kcSerial 3.0

Firmware User Guide

May 9, 2011

Wireless Data Communication Firmware
Supporting SPP, DUN & RFComm Bluetooth Profiles
With Remote Command & Control

Contents

1 Preface

kcSerial 3.0 firmware is a fully embedded RS-232 serial cable replacement application that provides point-to-point wireless communication and control between two Bluetooth devices using the Serial Port Profile (SPP), or Dial Up Networking Profile (DUN). This document describes major features of the firmware and a detailed reference for every command.

Our kcSerial 3.0 embedded Bluetooth Serial Port Profile firmware application is designed to operate KC Wirefree Bluetooth hardware modules with an easy to use AT style text command interface. Devices deploying kcSerial 3.0 firmware are designed to operate as a standalone wireless solution which does not require a host controller (PC or PDA). No additional software or drivers when connected to UART lines.

2 Command Interface

A kcSerial 3.0 device uses the UART interface by default. The kcSerial 3.0 firmware provides an embedded AT style text command interface. Upon power up or reset, the kcSerial 3.0 sends a startup message via UART, unless the Message feature has been disabled.

3 Compatibility

kcSerial 3.0 firmware offers fully qualified Bluetooth Serial Port Profile (SPP) and Dial Up Networking Profile (DUN), and can communicate with any Bluetooth device that supports those profiles. The kcSerial 3.0 user interface commands and several special features described in this document are unique to kcSerial 3.0 firmware.

kcSerial 3.0 is Bluetooth v2.1+EDR firmware for CSR chipsets designed to be compatible with all of our legacy firmware versions.

4 Features

- CommandMode – an interactive mode of operation that accepts easy to use AT style text commands for operation and configuration.
- RemoteMode – a remote command and control mode where any other Bluetooth serial device can remotely and wirelessly execute the kcSerial 3.0 AT Commands. This allows a kcSerial 3.0 unit to read, write digital and analog IO pins, and transfer data under remote control from any other Bluetooth device.
- BypassMode – a transparent mode that operates as a wireless serial cable. All data bytes sent to the module UART are transmitted wirelessly, and all incoming wireless data can be read from the module UART.
- Power conservation – deep sleep and sniff modes to minimize power consumption.
- UART interface – a 3 wire (TX, RX, Ground) or 5 wire (TX, RX, CTS, RTS, Ground) physical interface supports data rates from 1,200 bps to 3 Mbps.

5 Help Commands

5.1 Help Command

Enter `AT Help` from a terminal program to view a list of all kcSerial 3.0 AT Commands.

```
AT Help
-> [CommandList]
-> AioRead           AutoConnect       BatteryMon
-> BtAddr            Build             Bypass
-> CoD               ConfigRawBaud     ConfigUart
-> ConnDiscOverride  Connect           Connectable
-> ConnectDun        ConnectIos        ConnectScan
-> DebugMode         DeepSleep         Disconnect
-> DisconnectDun     Discoverable      DiscoverSvc
-> Discovery         DiscoveryRssi     EscapeCommand
-> FactoryReset      HciMode           HwFlowControl
-> InputCmdMode      InputConnect      InputSleepBlock
-> InquiryScan       LinkTest          LinkTimeout
-> LowLatency        Messages          Name
-> OutputActivity    OutputConnect     OutputCpu
-> OutputLowBatt     Pair              Pairable
-> PairingDelete     PairingOption     Passkey
-> PinCode           PioConfig         PioRead
-> PioSettings       PioStatus         PioStrong
-> PioWrite          RemoteCommand     Reset
-> RfcService        RfPower           RoleSwitch
-> Rssi              Security          SecurityAuth
-> ShowSettings      ShowStatus        Sniff
-> SniffSettings     SniffSubrate      SppService
-> Timer             TimerAio          TimerPio
-> Version           ZvMode
->
-> EscapeMode prefix: ~~~~~1
-> EscapeCmd  prefix: ~~~~~2
-> RemoteMode prefix: ~~~~~3
-> RemoteCmd  prefix: ~~~~~4
->
-> Command help: AT <command> ?
-> [EndCommandList]->
```

5.2 Other Helpful Commands

These helpful AT commands show configurations, firmware versions, device status, and other information.

```
AT Help, AT Build, AT PioSettings, AT ShowSettings, AT ShowStatus, AT Version
```

5.3 Command Parameters Help

For any AT Command, enter the command followed by ? to view the parameters accepted and expected by each command. An asterisk indicates an optional parameter. The Command Reference section has detailed information for each AT Command.

```
AT ConfigUart ?
-> AT ConfigUart <baudrate> <parity*> <stop*>
```

6 Command Summary

AT AioRead	Prints the voltage reading (in mV) of the Analog I/O pins 1,2,3.
AT AutoConnect	Enables AutoConnect feature and settings.
AT BatteryMon	Enables battery voltage monitoring using AIO 0.
AT BtAddr	Prints this Bluetooth device address.
AT Build	Prints detailed firmware edition information.
AT Bypass	Switches from CommandMode to BypassMode, when connected.
AT CoD	Changes the Class of Device. Saved in memory.
AT ConfigRawBaud	Sets non standard UART Baudrates.
AT ConfigUart	Sets the UART Baudrate, Parity, and StopBits. Saved in memory.
AT ConnDiscOverride	A manual override for Connectable and Discoverable setting when connected (normally both off).
AT Connect	Connect to a Bluetooth Spp Profile service of remote device.
AT Connectable	Enables incoming connections. Saved in memory.
AT ConnectDun	Connect to a Bluetooth Dun Profile service of remote device.
AT ConnectIos	Connect to the iPhone/iPad RfComm data service (Apple security chip required for data transfers).
AT ConnectScan	Configures incoming connection scanning settings. Saved in memory.
AT DebugMode	Enables internal debugging messages to assist with operational problems. Saved in memory.
AT DeepSleep	Enables Deep Sleep. Saved in memory.
AT Disconnect	Disconnect an existing Spp connection.
AT DisconnectDun	Disconnect an existing Dun connection.
AT Discoverable	Enables responses to incoming inquiry requests. Saved in memory.
AT DiscoverSvc	Discover Bluetooth profile services on a specified remote device.
AT Discovery	Search for Bluetooth devices.
AT DiscoveryRssi	Search for Bluetooth devices with RSSI reporting.
AT EscapeCommand	Enables escape character switching between CommandMode and BypassMode. Saved in memory.
AT FactoryReset	Restores all factory default settings in memory, and resets the device.
AT HciMode	Restarts device in Hci mode without kcSerial. Not saved.
AT HwFlowControl	Enables hardware flow control. Saved in memory.
AT InputCmdMode	Configures PIO switching between CommandMode and BypassMode. Saved in memory.
AT InputConnect	Configures PIO for connecting and disconnecting. Saved in memory.
AT InputSleepBlock	Configures PIO to block of deep sleep mode. Saved in memory.
AT InquiryScan	Configures inquiry scanning settings. Saved in memory.
AT LinkTest	BitErrorRate reading for a linked device.
AT LinkTimeout	Configures the link loss timeout parameter. Saved in memory.
AT LowLatency	Optimize data transfers for low latency or high throughput. Saved in memory.

AT Messages	Enables kcSerial system response messages. Saved in memory.
AT Name	Change the device name. Saved in memory.
AT OutputActivity	Configures PIO as Activity indicator. Can assign solid or blink modes. Saved in memory.
AT OutputConnect	Configures PIO as Connection indicator feature. Can assign solid or blink modes. Saved in memory.
AT OutputCpu	Configures PIO as Cpu indicator feature. Can assign solid or blink modes. Saved in memory.
AT OutputLowBatt	Configures PIO as Low Battery indicator feature. Can assign solid or blink modes. Saved in memory.
AT Pair	Executes a pairing attempt with a remote device. Pairing info saved in memory.
AT Pairable	Enables incoming pairing requests. Saved in memory.
AT PairingDelete	Deletes all paired device information from memory.
AT PairingOption	Set display and/or keypad capabilities for pairing procedures. Saved in memory.
AT Passkey	Command to send keypad entries during pairing procedures.
AT PinCode	Change the default PinCode. Saved in memory.
AT PioConfig	Configure the input/output direction of a PIO pin. Not saved.
AT PioRead	Read a PIO pin.
AT PioSettings	Prints all available PIO pins with state, i/o status, weak/strong status, & feature assignment.
AT PioStatus	Prints 16bit masks indicating state, i/o status, weak/strong status, hw availability.
AT PioStrong	Allows weak/strong configuration for a PIO pin.
AT PioWrite	Set a high/low value for a Pio pin. Not saved.
AT RemoteCommand	Enables RemoteCommand functionality.
AT Reset	Resets the device after 0.5 sec delay.
AT RfcService	Registers an RfComm service. Not saved.
AT RfPower	Configures the output power parameters. Saved in memory.
AT RoleSwitch	Switches the master/slave role of connected devices.
AT Rssi	Prints the Rssi readings of a connected device.
AT Security	Sets the security to level 1,2,3 (off, medium, high).
AT SecurityAuth	Restricts pairing to authenticated pairing only (level 3), and prevents automatic pairing.
AT ShowSettings	Prints the all of the device settings.
AT ShowStatus	Prints the operational state settings.
AT Sniff	Enables Sniff mode.
AT SniffSettings	Configure default Sniff mode settings. Saved in memory.
AT SniffSubrate	Sets Sniffsubrating configuration temporarily.
AT SppService	Enables the SPP profile. Save in memory.
AT Timer	Starts a general timer function.
AT TimerAio	Starts a general AIO reading timer function.
AT TimerPio	Starts a general PIO reading timer function.

AT Version	Prints the full version information, including edition.
AT ZvMode	Enables kcSerial 2.2 compatible host response messages.

7 Default Settings

The UART default setting is Baudrate 115200, 8 Data bits, No Parity, 1 Stop Bit.

The following settings are the default settings for kcSerial 3.0. These will all be restored, and saved, to these settings when the AT FactoryDefault command is issued.

```
AT ShowSettings
-> [Settings]
-> Name "kcSerial"
-> D AutoConnect 000000000000 0 0
-> D BatteryMon Low=0mV Off=0mV 0s
-> E Connectable
-> D DebugMode
-> D DeepSleep
-> E Discoverable
-> E EscapeCommand
-> D HwFlowControl
-> E LowLatency
-> E Messages
-> E Pairable
-> E RemoteCommand
-> D SecurityAuth
-> E Sniff
-> E SppService
-> D ZvMode
-> ClassOfDevice 001F00
-> ConnectScan 36 1024
-> InquiryScan 36 2048
-> LinkTimeout 5000
-> PairingOption Automatic
-> PinCode 1234
-> PrevConnect 000000000000
-> SecurityLevel 1
-> SniffSettings 0: Active 0 0 0 0 2
-> SniffSettings 1: Sniff 32 200 1 16 30
-> SniffSettings 2: Sniff 160 640 1 16 0
-> Uart 115200-8-N-1
-> [EndSettings]
```

8 AT Command Syntax

kcSerial 3.0 is a unique and proprietary AT style command interface by KC Wirefree that provides an extensive command language for easily configuring many device settings, and managing connections, disconnections, and operating the PIO (Programmable Input Output) pins. The AT Commands interact with flash memory storage to save many configurations and settings. All kcSerial 3.0 devices use flash memory for storage, and consequently, no settings are permanent, but rather stored in persistent flash memory, including kcSerial firmware itself. Please refer to the Firmware upgrade section for details regarding firmware updating.

8.1 UART And USB Interface

kcSerial 3.0 does not currently provide the AT Command interface using USB directly. However, a USB-UART bridge chip may be used to provide the AT Command interface. The bridge chip will physically connect to the USB port on the PC to the USB bridge chip. The USB device drivers are provided by the bridge chip manufacturer, and will provide a virtual UART COM port. The bridge chip will physically connect to the kcSerial device using UART lines. Thus, AT Commands can be entered using the virtual COM port, and will automatically transferred to the kcSerial UART lines via the bridge chip and USB cable. Suitable USB-UART bridge chips are available from SiliconLabs and FTDI.

8.2 Command Entry

This Firmware User Guide provides instructions for using the standard kcSerial 3.0 AT Command interface via the default UART interface. AT Commands are entered as characters (bytes) via the UART interface. The character is used in this Firmware User Guide to indicate an <enter> key press that generates the EOL (end of line) indication, which may or may not include a line feed, depending on terminal software settings used. kcSerial 3.0 will accept either <CR> or <CR><LF> characters to denote the EOL (end of line). Responses will be prefixed with the -> characters, and will include <CR><LF> as the EOL indicator.

```
AT Version<CRLF>          [Hex: 41 54 20 56 65 72 73 69 6F 6E 0D 0A]
-> kcSerial 3.0<CRLF>     [Hex: 2D 3E 20 6B 63 53 65 72 69 61 6C 20 33 2E 30 0D 0A]
```

8.3 Case Insensitive Commands

All kcSerial AT Commands and parameters (except for changing the device Name parameter) are converted to lower case. So, kcSerial AT Commands are case insensitive. This Firmware User Guide illustrates commands using upper case characters simply to make these commands easily read and understood.

8.4 Proper Case Output Messages

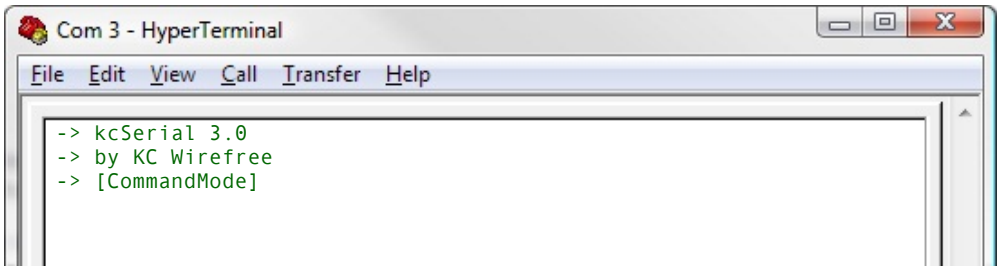
Output messages are fixed in firmware except for variable parameters, and are output via UART with the upper and lower cases intact.

9 Getting Started

Physically connect the kcSerial 3.0 device UART pins to a PC serial port. Modules require voltage shifting from TTL to match RS-232 voltage levels. Start a terminal program, such as HyperTerminal, and configure the port settings.

▲Default UART settings in kcSerial 3.0 are 115200 bps, 8 data bits, no parity, 1 stop bit, no flow control.

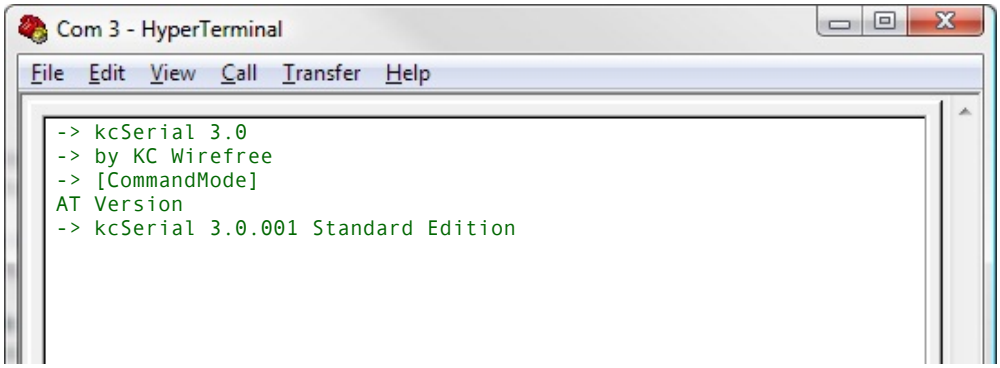
Open the PC COM port with HyperTerminal, and power on or reset the kcSerial 3.0 device to see the welcome message.



Syntax: All kcSerial output messages have the -> prefix, and have a 2 byte <CRLF> End Of Line marker. Also, AT commands are case insensitive. We use capital letters for legibility.

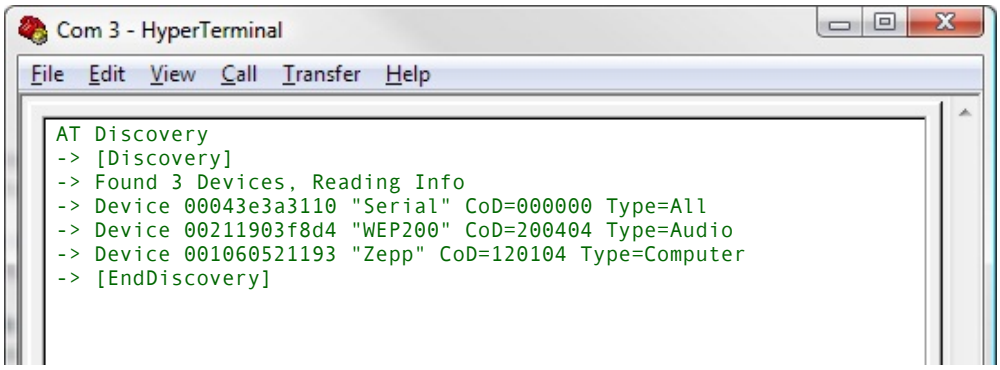
Try the AT Version command. kcSerial 3.0 accepts simple text commands as byte streams followed by the <CR> or <CRLF> to indicate the EOL. The <CR> is received by simply pressing the <Enter> key. Terminal software may add the <LF> LineFeed character after pressing the <Enter> key, depending on the terminal software settings. Enter the Version command:

```
AT Version<CRLF>
```



Next, find any available Bluetooth devices. Remember, Bluetooth devices must be in “Discoverable” mode in order to find them in a search. kcSerial 3.0 is always in Discoverable and Connectable mode when not actively connected to another device. Enter the AT Discovery command:

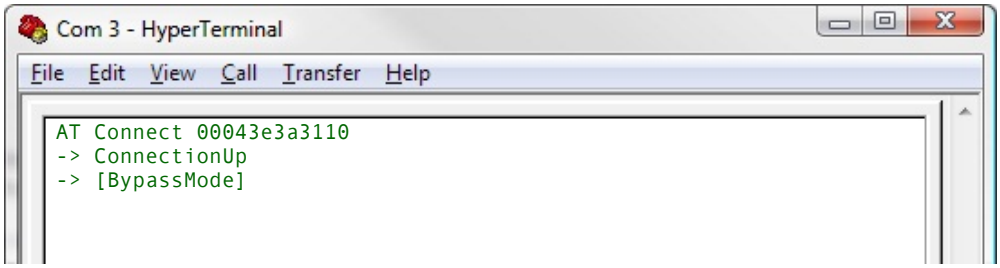
```
AT Discovery<CRLF>
```



kcSerial 3.0 will output several messages when initiating a search for devices. After a complete search, the total number of devices found will be indicated. Next, kcSerial 3.0 will connect to each device individually and retrieve it's information including the btaddress, device name, and class of device.

Connecting to another serial device may require a pin code, or bonding in order to connect, based on the security settings of each device. Obviously, this can get tricky depending on the level of security, and individual settings intended to prevent some connections. Let's try a simple, no security connection. kcSerial 3.0 has security disabled by default. To connect, issue this command: AT Connect 00043e3a3110.

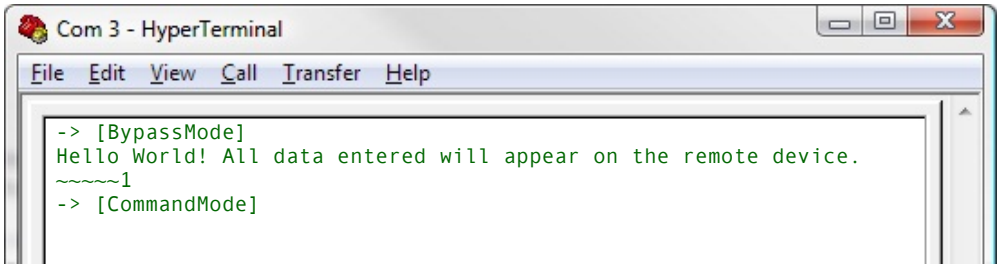
```
AT Connect 00043e3a3110<CRLF>
```



Upon successful connection, kcSerial 3.0 immediately switches into BypassMode. Any data received on the local UART will be transmitted wirelessly to the remote device. Also, any data received wirelessly from the remote device will be sent to the UART and received by the local serial port. The BypassMode is a completely transparent transfer mode that operates as a serial cable – with one exception.

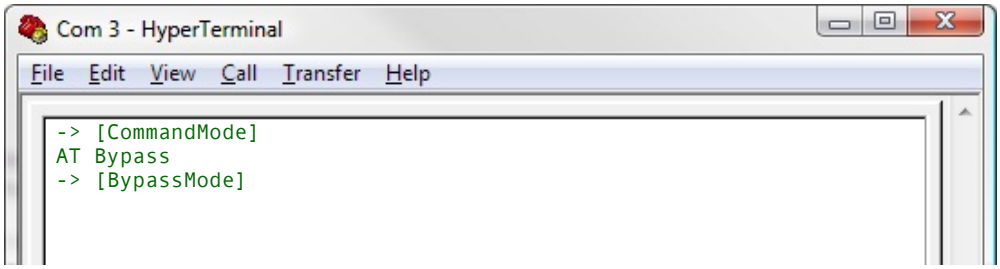
You can escape from BypassMode by issuing the EscapeMode character sequence. Note: no EOL marker is necessary for the escape command. Also note, this escape command must be sent at one time, in a single data packet. kcSerial cannot delay valid, wireless data transmissions while waiting to process individual character keystrokes. Hyperterminal users must send a "text file" or use a key macro containing the escape sequences.

```
~~~~~1
```



The escape characters will not be transmitted to the remote device, nor will any characters or data received from the local serial port while in CommandMode. When done using CommandMode, you can return to BypassMode with the following command:

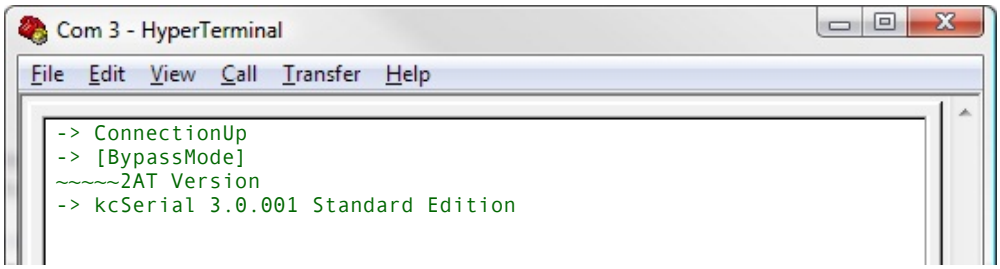
```
AT Bypass<CRLF>
```



If kcSerial 3.0 is not currently connected to a remote device, the Bypass command will fail, and remain in CommandMode.

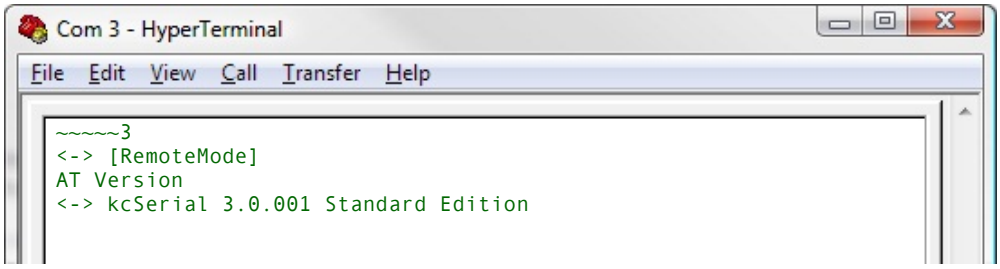
From BypassMode, you can also enter a single AT Command, without switching modes. Try issuing the AT Version command directly from BypassMode. By using the EscapeCmd prefix, the kcSerial will not send the data bytes, but rather execute any valid command instead.

```
~~~~~2AT Version<CRLF>
```



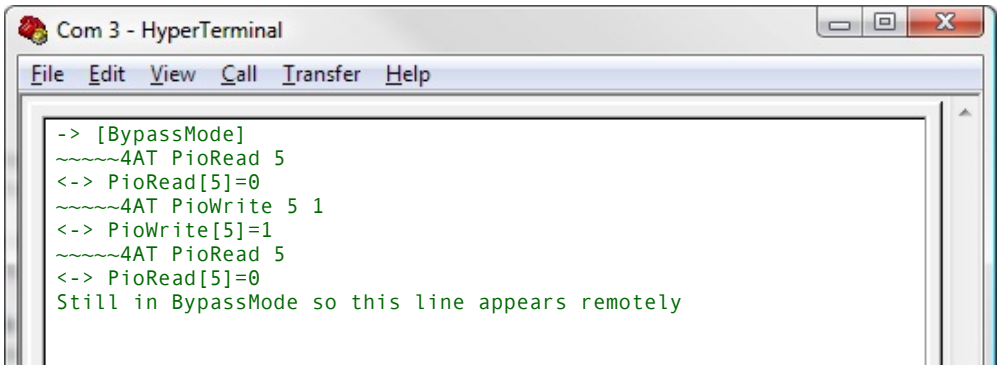
While currently connected, issue the RemoteMode sequence. The remote device must be a kcSerial device, but the local device can be from any manufacturer. The local device sends the following characters, and the remote kcSerial device responds with a RemoteMode prompt. Now, any characters send to the remote kcSerial device will be interpreted as AT Commands. An EOL marker is optional, which will be discarded, along with any other trailing characters.

```
~~~~~3
AT Version<CRLF>
```



Additionally, a RemoteCmd sequence can be sent for a quick operation, without changing modes on the remote kcSerial device. An EOL marker is necessary following the AT Command. System responses are sent wirelessly and have the <-> prefix, indicating a remote response.

```
~~~~~4AT PioRead 5<CRLF>
~~~~~4AT PioWrite 5 1<CRLF>
~~~~~4AT PioRead 5<CRLF>
```



10 Modes of Operation

10.1 HCI Mode

The highest level command interface defined by Bluetooth SIG is the Host Controller Interface (HCI), which is typically used by Bluetooth stack software operating on a PC. HCI mode is very complex, and not intended as a human interface. However, in addition to Bluetooth stack software, there are several manufacturer related applications that require the HCI interface, including a Flash Memory configuration program, Firmware Installation program, and Device Test program. While these programs are not usually required, they available from KC Wirefree upon request.

kcSerial 3.0 is capable of providing raw HCI level access via UART using the `AT HciMode` command. The kcSerial device will reboot into HciMode using the UART interface at the default settings, 115200-8-N-1. With the kcSerial device in this mode, our kcSerial embedded firmware is not used, but instead, standard PC based Bluetooth stack software must take control. The kcSerial device in HciMode will operate similar to a common Bluetooth USB dongle, except that the UART interface must be utilized with a kcSerial device.

10.2 CommandMode

The CommandMode is the default mode when not wirelessly connected. This interactive mode accepts the AT commands from the local device UART port for operational and configuration control. Connections can be initiated from this mode. Also, this mode can be entered from BypassMode while wirelessly connected in order to execute commands, and a simple AT command (`AT Bypass`) can switch the kcSerial 3.0 device back to BypassMode.

10.3 BypassMode

The BypassMode is the default mode when wirelessly connected, allowing completely transparent data traffic between Bluetooth devices. In this data transfer mode all data bytes received from the local UART pins are simply transmitted to the remote device. Also, any data bytes received wirelessly, are sent for output to the local UART pins.

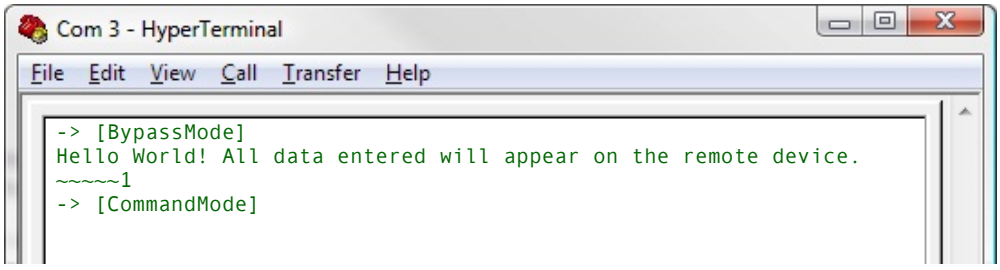
However, if the EscapeCommand feature is enabled, the data stream is monitored for special EscapeMode and EscapeCmd character sequences received from the local UART port.

10.4 EscapeMode Sequence

The EscapeMode sequence used to switch from BypassMode to CommandMode during an active connection is:

```
~~~~~1
```

Note, escape commands must be sent at one time, in a single data packet. kcSerial cannot delay valid, wireless data transmissions while waiting to process individual character keystrokes. Hyperterminal users must send a “text file” or use a key macro containing the escape sequences.

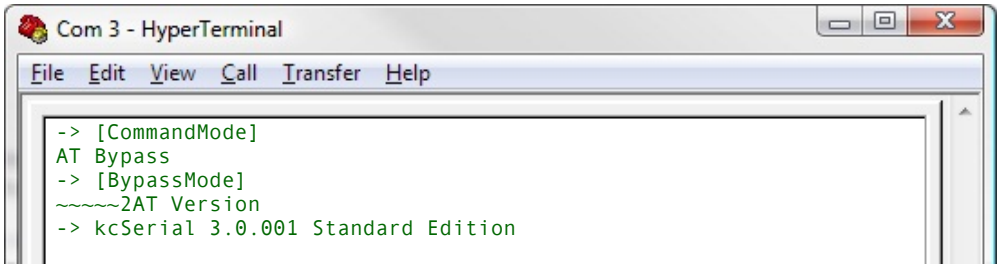


If the EscapeMode sequence is detected, the local kcSerial 3.0 device will switch out of BypassMode and into CommandMode. The EscapeMode prefix does not require an EOL marker, and any characters following the six character sequence are ignored. Use the AT Bypass command to switch back to CommandMode. Any data received by the kcSerial device while in CommandMode, are quietly ignored.

10.5 EscapeCmd Prefix

The EscapeCmd prefix is used to issue AT Commands while remaining in BypassMode is:

```
~~~~~2
```



If the EscapeCmd sequence is detected, the local kcSerial 3.0 device will interpret the following text as an AT Command. Any characters or data following the special sequence will be interpreted as commands by CommandMode, and an EOL marker is required for proper interpreting of the AT Command.

10.6 RemoteMode

RemoteMode is a unique and powerful remote control mode only available from KC Wirefree. With a wireless connection, the RemoteMode escape sequence can be sent to a remote kcSerial 3.0 device, by any other Bluetooth device. If RemoteCommand is enabled, the kcSerial device will switch into RemoteMode. RemoteMode is identical to CommandMode, except that the AT commands are received wirelessly. Also, any command response messages are sent wirelessly.

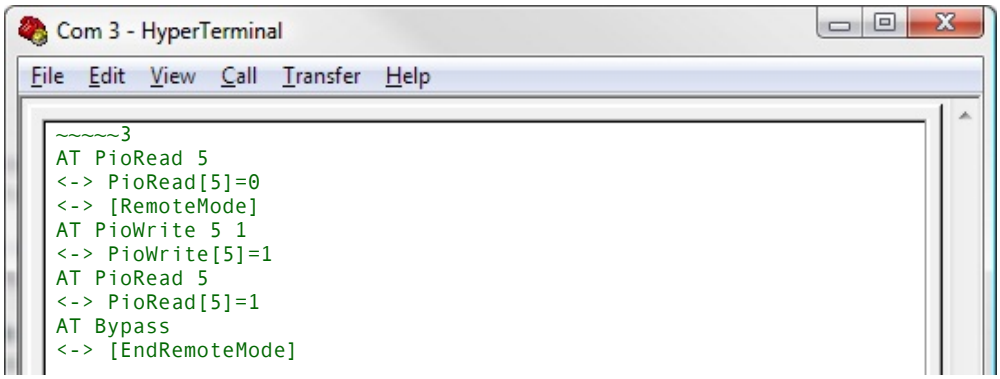
A simple AT command (AT BypassMode) can be sent wirelessly to the device in RemoteMode, causing the device to switch back into BypassMode. Note, the local Bluetooth device must be in BypassMode or an equivalent data transfer mode, so that the character sequences representing AT commands are sent wirelessly to the remote kcSerial 3.0 device.

A kcSerial device monitors the incoming wireless data stream for the special RemoteMode sequence and RemoteCmd prefix if the RemoteCommand feature is enabled.

10.7 RemoteMode Sequence

RemoteMode sequence sent by the local device to place the remote kcSerial device in RemoteMode is:

```
~~~~~3
```

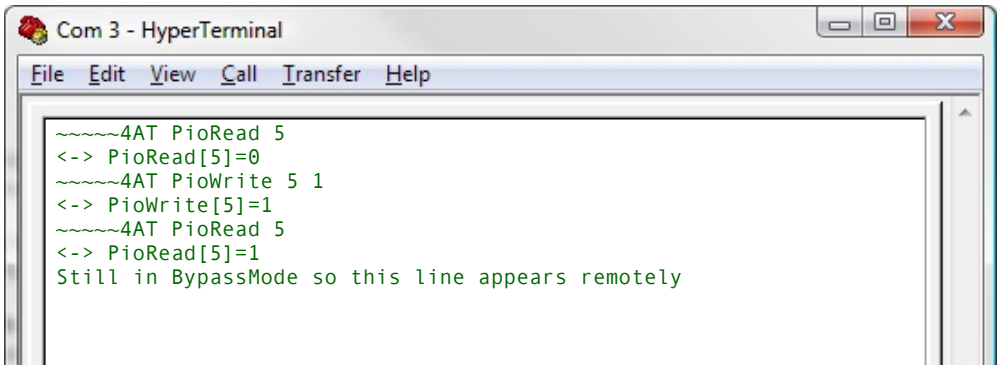


10.8 RemoteCmd Prefix

The RemoteCmd prefix used to issue AT Commands to a remote kcSerial device is:

~~~~~4

The RemoteCmd sequence is a prefix that can be sent preceding an AT Command. A remote kcSerial device will interpret the following AT Command, and reply wirelessly. The RemoteCmd sequence is intended to quickly send single commands and receive replies without entering RemoteMode. The remote kcSerial device must have its RemoteCommand feature enabled.


~~~~~

11 PIO Features

Several special features are available using Peripheral Input Output (PIO) pins.

11.1 OutputActivity

A simple PIO output feature that is activated when kcSerial 3.0 is sends or receives wireless data. The output indicator action runs for 200ms, and is not related to the amount of data transferred. The OutputActivity feature can be disabled or configured to a specified PIO pin, with a choice of solid, inverted, or blinking actions.

11.2 OutputConnect

A simple PIO output feature that is activated when kcSerial 3.0 is wirelessly connected to another Bluetooth device. The OutputConnect feature can be disabled or configured to a specified PIO pin, with a choice of solid, inverted, or blinking actions.

11.3 OutputCpu

A simple PIO output feature that is activated when kcSerial 3.0 is on. The OutputCpu feature can be disabled or configured to a specified PIO pin, with a choice of solid, inverted, or blinking actions. This Output will go LOW when the BatteryMonitor feature is enabled, and the shutoff threshold voltage has been reached.

11.4 OutputLowBatt

A simple PIO output feature that is activated when kcSerial 3.0 detects a low battery voltage, as specified by the AT BatteryMon command.

11.5 Output Feature Blink Settings

The blink configuration available for each Output feature:

0 = SolidLow	A reverse logic output where LOW indicates feature is ON, and HIGH indicates feature is OFF.
1 = SolidOn	A standard logic output where HIGH indicates feature is ON, and LOW indicates feature is OFF.
2 = SlowBlink	A blinking output. 300ms HIGH, 300ms LOW.
3 = FastBlink	A blinking output. 50ms HIGH, 50ms LOW.
4 = BlipOn	A brief blink output. 30ms HIGH, 2970ms LOW.
5 = LowPwrBlip	Minimum power blinking output. 30ms HIGH, 9970ms LOW.

12 InputCmdMode

A PIO input feature that causes kcSerial 3.0 to switch between BypassMode and CommandMode. This PIO feature allows switching to CommandMode even when the EscapeCommand setting has been disabled. Disabling the EscapeCommand allows the fastest data throughput, by not checking for the EscapeMode character sequence. kcSerial 3.0 will toggle between CommandMode and BypassMode whenever this input changes from LOW to HIGH. If the device is not wirelessly connected, then it cannot switch to BypassMode.

12.1 InputConnect

A PIO input feature that causes kcSerial 3.0 to initiate a connection. The first choice, is to re-start any AutoConnect configuration. When AutoConnect is not configured, kcSerial 3.0 will attempt to connect to the previously connected device. Additionally, if the device is currently connected, this feature will disconnect. kcSerial will initiate the connection or disconnection when this input changes to HIGH.

12.2 InputSleepBlock

A PIO input feature that causes kcSerial 3.0 to block or allow deep sleep mode. DeepSleepMode must be enabled, which will allow the device to go into DeepSleepMode whenever possible. When this input feature is enabled, a HIGH signal on this pin will wake the device and prevent deep sleep, while a LOW signal will allow deep sleeping when possible. The InputSleepBlock feature can be disabled or assigned to any PIO pin.

13 Auto Connect Feature

This feature provides automatic connections for cable replacement application. AutoConnect is simply SmartCable renamed from previous versions of kcSerial. The following AT Commands are used for AutoConnect implementation:

```
AT AutoConnect <e/d> <btaddr> <attempts> <interval>
AT InputConnect <enable> <pio>
```

13.1 AutoConnect

To enable the AutoConnect feature, enter the `AT AutoConnect` command with all parameters. When the AutoConnect feature is enabled, kcSerial 3.0 will start the AutoConnect connection attempts upon power up, reset, or dropped link. Additionally, the PinConnect feature can be enabled, which will manually restart the AutoConnect connection sequence whenever the assigned PIO pin is triggered High.

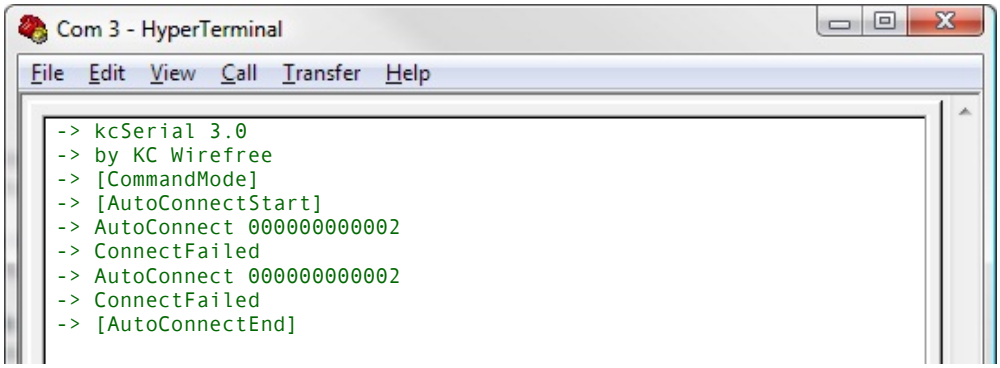
The AutoConnect feature with 0 attempts specified, is still available for a single manual connection attempt with the InputConnect feature. The current AutoConnect settings can be seen with the `AT ShowSettings` command.

Delete the AutoConnect feature and settings by entering only a 0 parameter.

```
AT AutoConnect D
```

Example AutoConnect setup:

```
AT AutoConnect E 000000000002 2 50
AT Reset
```



14 Power Saving Features

kcSerial 3.0 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.

14.1 Deep Sleep Mode

In kcSerial 3.0, 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 32 KHz oscillator instead. When enabled, DSM automatically enters this halt state whenever possible. Scheduled CPU activity, PIO interrupts, and UART requests will automatically resume active mode operation.

14.2 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, an external controller must pulse any PIO pin or the device's CTS line and wait 10ms for the device to wake.

14.3 InputSleepBlock

kcSerial 3.0 supports a Deep Sleep Blocking feature using a designated PIO (can be assigned to any PIO). When enabled, an HIGH signal on the PIO will temporarily block Deep Sleep Mode. Normal DSM operation will resume when this signal is PIO is LOW.

14.4 Sniff

kcSerial 3.0 supports Sniff Mode connections by default. The power savings gained with the Sniff feature can be higher when used in conjunction with Deep Sleep Mode. Sniff will reduce data throughput, and increase latency. Sniff mode parameters can be configured with the AT SniffSetting command.

15 Security

Bluetooth v2.1 introduces many changes to the pairing procedure, and provides several new methods in addition to the legacy Pincode entry (also known as Passkey). The changes are intended to provide two benefits: allow easier pairing overall, and prevent unseen strangers from pairing to your device. The legacy Pincode entry method does not prevent unseen users from pairing with your device because they can simply guess the Pincode, where the Pincode is usually either 0000 or 1234. The new pairing methods include provisions for numeric comparison, simple yes/no acceptance, and a more robust Pincode entry method. Even the automatic pairing acceptance is more secure, to prevent malicious middle-man devices from intercepting transmissions during pairing and communications.

The following commands are all directly related to security:

`Connectable, Discoverable, Pair, Pairable, PairingDelete, PairingOption, Passkey, PinCode, Security, SecurityAuth`

▲ kcSerial 2.2 and v2.4 compatibility: The most of the security related commands have changed names and extended functionality.

15.1 Connectable, Discoverable, Pairable

Basic features that can be enabled or disabled.

All incoming pairing requests will be rejected by disabling pairing using the `AT Pairable` command.

All incoming connection requests by remote devices will be rejected by disabling connections using the `AT Connectable` command.

All discovery requests by remote devices will be rejected by disabling discovery using the `AT Discoverable` command.

15.2 New Pairing Methods

Bluetooth v2.1 now provides four new pairing methods. These are only available when both devices are using Bluetooth v2.1 or higher. If one of the devices is using a legacy version of Bluetooth, then pairing will default to the legacy method of providing a PinCode which is typically a four digit number.

Bluetooth v2.1 uses a lookup matrix table to determine which pairing method is used. The pairing method is determined by comparing the input and output capabilities that are broadcast by each device. Simply, each device will indicate whether it has a keypad and/or a display, the then an appropriate pairing method is determined.

- Numeric Comparison - where a random number is shown on one device, and the other device must manually enter and reply with this displayed number.
- Passkey Entry – where each device enters and sends identical Passkeys. This is similar to the legacy method of 4 digit Passkeys, but now allows 16 alpha numeric digits.
- Yes/No Confirmation – where at least one device must enter and send a simple yes/no response to a pairing request.
- Just Works – where all pairing requests are accepted automatically. A bond and link key is created and saved, but the pairing is not considered "authenticated". While this seems insecure, it improves security regarding the interception of transmissions by a middle-man device. For actual secure pairing, one of the other, interactive "authentication" methods must be used.

15.3 Pairing Options

Setting the pairing options allow configuration of Bluetooth 2.1 pairing behavior. The options include specifying input capability and output capability. Bluetooth 2.1 will determine which pairing method is to be used depending on the input/output capabilities of both devices attempting to pair. kcSerial provides the `AT PairingOption` command to configure these capabilities. The kcSerial default PairingOption is 3. Available Pairing Options:

- 0 = DisplayOnly
- 1 = Display+Keypad
- 2 = KeypadOnly
- 3 = Automatic (No Keypad, No Display)
- 4 = Reject (No Bluetooth v2.1 Pairing Options)

DisplayOnly

When using the PairingOption of DisplayOnly, any characters that need to be display will simply print as a system message to the UART. These characters can be read and displayed by a terminal application, or read by a microprocessor.

Display+Keypad

When using the PairingOption of Display+Keypad, any characters that need to be display will simply print as a system message to the UART. These characters can be read and displayed by a terminal application, or read by a microprocessor. Additionally, keypad responses must use the AT Passkey command to reply with keypad entries.

KeypadOnly

When using the PairingOption of KeypadOnly, keypad responses must use the `AT Passkey` command to reply with keypad entries.

Automatic

When using the PairingOption of Automatic, the only pairing method available will be the JustWorks method, which is automatic pairing. This is not a secure method, and is pairing made by this method are marked as Not Authenticated.

Negotiated Pairing Method

	A: DisplayOnly	**A: Display+Keypad**	**A: KeypadOnly**	**A: Automatic**
B: DisplayOnly	Numeric Comparison Auto confirm on A Auto confirm on B [Not Authenticated]	Numeric Comparison Confirm on A, Auto confirm on B [Not Authenticated]	Passkey Entry Display on B Input on A [Authenticated]	Numeric Comparison Auto confirm on A Auto confirm on B [Not Authenticated]
B: Display+Keypad	Numeric Comparison Auto confirm on A Confirm on B [Not Authenticated]	Numeric Comparison Confirm on A Confirm on B [Authenticated]	Passkey Entry Display on B Input on A [Authenticated]	Numeric Comparison Auto confirm on A Confirm on B [Not Authenticated]
B: KeypadOnly	Passkey Entry Display on A Input on B [Authenticated]	Passkey Entry Display on A Input on B [Authenticated]	Passkey Entry Input on A Input on B [Authenticated]	Numeric Comparison Auto confirm on A Auto confirm on B [Not Authenticated]
B: Automatic	Numeric Comparison Auto confirm on A Auto confirm on B [Not Authenticated]	Numeric Comparison Confirm on A Auto confirm on B [Not Authenticated]	Numeric Comparison Auto confirm on A Auto confirm on B [Not Authenticated]	Numeric Comparison Auto confirm on A Auto confirm on B [Not Authenticated]

15.4 Authentication Required

The `AT SecurityAuth` command can be used to enabled or disabled a mandatory authentication requirement. Note: if mandatory authentication is enforced, then pairing will always fail with security levels 0-2. Furthermore, the automatic pairing method cannot be used, as this will not generate an authenticated pairing. Enabling the mandatory authentication requirement is only recommended for very high security applications.

15.5 Pair Command

The `AT Pair` command simply initiates pairing without making a subsequent connection. Note: initiating a connection when devices have not previously paired, will also initiate pairing, just like the `AT Pair` command, but will additionally establish a connection following successful pairing.

15.6 Security Command

The AT Security command has a level parameter.

Security Level 0

All security is off.

Security Level 1

Level 1 is no security, no encryption. When in level 1 security mode, no outgoing pairing requests will be made, but any incoming pairing requests by the remote device will be accepted and handled as needed. Encryption is not initiated, but is used when requested by the other device.

Security Level 2

Level 2 security requests pairing and accepts all requests automatically using the new Bluetooth v2.1 "just works" simple pairing method, and saves pairing keys and devices in the paired list. Automatic pairing in level 2 security is considered "bonded", but not "authenticated", when automatic pairing methods are used. Level 2 security will not be sufficient if the remote unit demands authenticated pairing (level 3 security). The legacy Pincode pairing method used by Bluetooth v1.2 and v2.0 is implemented and is completely backward compatible when requested by a legacy device. However, legacy pairing is automated, and kcSerial can only respond with the stored Pincode.

Security Level 3

Level 3 security provides encryption and only allows authenticated pairing of devices, which occurs when an interactive method is used to complete pairing. Note: automatic confirmations on either device, generate Non Authenticated pairings. This is the highest levels of standard Bluetooth security available, and requires user interaction, which means a display and or keypad is necessary to perform pairing. In kcSerial 3.0, all pairing information needed for display is sent via UART to be further implemented. Additionally, the AT Pincode command is used to send keypad entry information to the other device tom complete pairing procedures. A keypad device must be implemented for embedded applications, and able to provide the AT Pincode command via UART.

16 kcSerial Compatibility Notes

16.1 Compatibility Notes for kcSerial 2.2

The old input command prefixes of AT+ZV and AT+KC are still recognized in addition to the new, simplified AT prefix. In order to receive the old kcSerial 2.2 output message prefix of AT-ZV, the new ZvMode setting must be enabled. Otherwise, all command messages are sent with the new "-> " prefix.

The following kcSerial 2.2 commands are discontinued in kcSerial 3.0: Hold, Park, RemoteCmdDisconnect, StreamingSerial.

The StreamingSerial feature (enabling or disabling hardware flow control using the CTS/RTS lines) is still available, but not via the previous AT StreamingSerial command. kcSerial 3.0 must have hardware flow control setting enabled or disabled directly in the flash memory using an external software tool. Hardware flow control is enabled by default. If no flow control is desired, contact us for instructions, or simply pull the CTS line LOW, and leave the RTS line floating for 3-wire serial applications using slower baud rates.

16.2 Compatibility Notes for kcSerial 2.4

The following kcSerial 2.4 AT Verbose command is renamed DebugMode, however Verbose is still accepted.

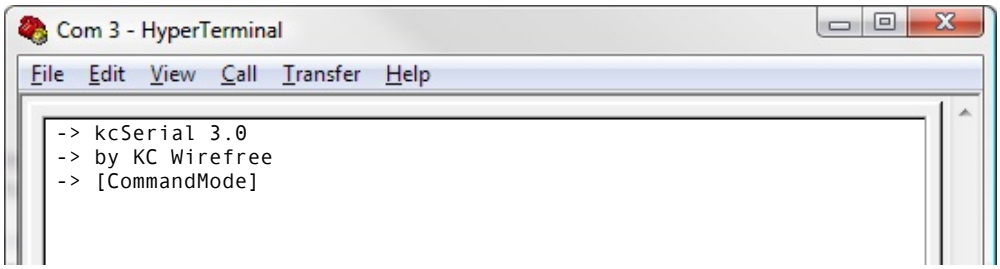
17 Firmware Updates

kcSerial firmware can be updated via UART by using our kcDFUWizard program on a PC computer. This program is available online http://www.kcwirefree.com/downloads.html

18 Command Reference

18.1 Welcome Message

The initial messages are sent upon power up or reset when Messages are enabled (the default setting).



18.2 AT AioRead

The AioRead command is used to read an analog voltage level on the Aio0 or Aio1 pins. The level is printed in millivolts, and is capable of reading between 0 and 1800mV.

Command `AT AioRead <aio>`

<aio> `0=Aio0, 1=Aio1`

Example
```
AT AioRead 0
-> AioRead[0]=1250mV
```

18.3 AT AutoConnect

The AutoConnect command provides automatic connections. A remote device address is specified along with the number of connection attempts, and the interval between attempts. A value of 0 <attempts> will simply store the connection information without automatically connecting. However, one manual connection attempt will be started when using the AT AutoConnect feature. When enabled, AutoConnect will attempt to automatically connection to the specified device upon startup or link loss. Issuing a disconnect command from the local device, will disable the AutoConnect feature until reset, or manually enabled again. This feature is saved in memory, and can be viewed using the AT ShowSettings command.

Command `AT AutoConnect <e/d> <btaddr> <attempts> <interval>`

<e/d> `e=Enabled, d=Disabled`
<btaddr> `The 12 digit Bluetooth address of the remote device`
<attempts> `The number of reconnection attempts (1000 = unlimited)`
<interval> `The wait interval in seconds between attempts`

Example
```
AT AutoConnect D
-> AutoConnect Disabled
```

Example `AT AutoConnect E 0123456789AB 10 30`

```
-> AutoConnect attempts[10] interval[30]sec
```

18.4 AT BatteryMon

The BatteryMon command provides battery voltage monitoring. It offers a low battery warning period, and a shutoff limit. Battery monitoring requires an external voltage divider circuit and use of AIO_0. This pin is reads voltages 0-1800mV with 256 discrete readings (±7mV). There is no de-bouncing, so readings may intermittently trigger the low battery level. A low battery state will set the OutputLowBatt Pio HIGH, if this feature is enabled, or output message to the local UART for every reading that is low. The OutputCpu Pio will be set LOW, when this feature is enabled, and a local UART message will be generated for every reading below the shutoff level. This feature is saved in memory, and can be viewed using the AT ShowSettings command.

Command `AT BatteryMon <e/d> <low mv*> <off mv*> <period*>`

<e/d> `e=Enabled, d=Disabled`
<low mv> `low battery warning level in mV`
<off mv> `shutoff battery level in mV`
<period> `seconds between voltage readings`

Example
```
AT BatteryMon E 1000 800 30
-> BatteryMon Enabled Low=1000mV Off=800mV Period=30s
```

Example
```
AT BatteryMon D
-> BatteryMon Disabled
```

Notification `-> [LowBatteryWarning]`

Notification `-> [LowBatteryShutoff]`

18.5 AT BtAddr

This command returns the Bluetooth device address for this device.

Command `AT BtAddr`

Example
```
AT BtAddr
-> 0123456789AB
```

18.6 AT Build

This command returns the version, edition, build number, build date and time, hardware platform, Bluetooth version, and Copyright information for the device. The edition information is either Standard, for KC Wirefree standard editions, or may contain a custom edition name for individual customers with customized firmware and/or default settings.

Command `AT Build`

Example `AT Build`

```
-> [BuildInfo]
-> BTAddress: 000000000001
-> Bluetooth: v2.1+EDR
-> Hardware:  KC-21
-> Firmware:  kcSerial 3.0
-> Build:     035
-> Edition:   Standard
-> Date:      Oct 29 2010 16:36:29
-> Copyright: 2010 KC Wirefree Corporation
-> [EndBuildInfo]
```

18.7 AT Bypass

The Bypass command is used to return the kcSerial 3.0 device to BypassMode from CommandMode or RemoteMode, when the device is still wirelessly connected. When the device is not actively connected, BypassMode is unavailable, and this command will fail.

Command `AT Bypass`

Example
```
AT Bypass
-> [BypassMode]
```

Example
```
AT Bypass
-> ErrNotConnected
```

18.8 AT CoD

The CoD command changes the Class of Device setting, which is reported to remote devices inquiring this device. The kcSerial default value is 0x001F00 which indicates an all-purpose data device. Serial devices are not specifically defined by Bluetooth, as the Serial connection capabilities are used by most other profiles. This feature is saved in memory, and can be viewed using the AT ShowSettings command.

Command `AT CoD <codval>`

<codval> `A 24-bit hexadecimal Bluetooth CoD value.`

Example
```
AT CoD 42E20A
-> ChangedCoD 42E20A
```

18.9 AT ConfigRawBaud

The ConfigRawBaud command can configure the baud rate settings to non-standard rates. This feature is saved in memory, and can be viewed using the AT ShowSettings command.

Command `AT ConfigRawBaud <baudrate>`

<baudrate> `any`

Example
```
AT ConfigRawBaud 32100
-> ConfigRawBaud 32100
```

18.10 AT ConfigUart

The ConfigUart command can configure the UART settings, including Baudrate, Parity, and number of Stop Bits. This setting is saved in memory. The response message will be sent, and then the baud rate will be updated. The number of Data Bits is 8 and is not configurable. The parity and stop parameters are optional, and will remain unchanged if not specified.

Default UART settings in kcSerial 3.0 are 115200 bps, 8 data bits, no parity, 1 stop bit, no flow control.

Note: The UART buffer can become corrupted following a baud rate change. Sometimes the next command issued at the new baud rate has corrupt leading characters or data.

▲ kcSerial 2.2 and kcSerial 2.4 compatibility: The ChangeBaud and ChangeDefaultBaud commands are still accepted, and will evoke this command. The ChangeBaud command is a temporary unsaved baudrate change.

Command `AT ConfigUart <baudrate> <parity*> <stop*>`

<baudrate> `9600,19200,38400,57600,115200,230400,460800,921600,1382400`
<parity> `None, Odd, Even`
<stop> `1 or 2`

Example
```
AT ConfigUart 115200
-> ConfigUartOk 115200-8-n-1
```

Example
```
AT ConfigUart 115200 even 2
-> ConfigUartOk 115200-8-e-2
```

18.11 AT ConnDiscOverride

The ConnDiscOverride command can enabled the device to remain Connectable and/or Discoverable even when connected. Normally, the device will not be either Connectable or Discoverable while connected to a remote device. Note, remaining Connectable while already connected, does not provide additional or multiple connection abilities at this time. Future editions may allow additional connections. This feature is saved in memory, and can be viewed using the AT ShowSettings command.

Command `AT ConnDiscOverride <e/d> <e/d>`

<e/d> `Connectable Override e=Enabled, d=Disabled`
<e/d> `Discoverable Override e=Enabled, d=Disabled`
Example
```
AT ConnDiscOverride d e
-> ConnDiscOverride Disabled Enabled
```

18.12 AT Connect

The Connect command is used to initiate a connection with the specified device. Upon successful connection, kcSerial 3.0 immediately switches into BypassMode. If no Bluetooth device address is provided, then the address of the last connected device will be used. If no previous connection exists, then connection will fail when an address is not provided.

Additionally, if an optional RfComm Channel parameter is provided, the connection will be made to the service assigned to the indicated channel on the remote device. Alternatively, if an optional UUID parameter is provided, then kcSerial will search for the specified UUID service available on the remote device, and attempt to connect to the assigned RfComm channel on the remote device. Neither the optional Channel or UUID service will be used when the Connect command is issued without any parameters, thus using the last connected device. If the last connected device was a custom UUID or Channel, the connection will fail.

▲ kcSerial 2.2 and kcSerial 2.4 compatibility: The improves and replaces the AT+ZV SppConnect command.

Command `AT Connect <btaddr*> <channel/uuid*>`

<btaddr*> `The 12 digit Bluetooth address of the remote device`
<channel*> `The remote RfComm service channel number`
<uuid*> `The remote RfComm service UUID`

Example
```
AT Connect
-> ConnectionUp
-> [BypassMode]
```

Example
```
AT Connect 0123456789AB
-> ConnectionUp
-> [BypassMode]
```

Example
```
AT Connect 0123456789AB 5
-> ConnectionUp
-> [BypassMode]
```

Example
```
AT Connect 0123456789AB 1104
-> ConnectionUp
-> [BypassMode]
```

Failure Messages
```
-> ConnectFailed
-> ConnectFailed NoAddress
-> ConnectServiceFail
-> ConnectRejected NotConnectable
-> ConnectRejected ByRemote
```

ConnectFailed Failed – generic connection failure.

ConnectFailed NoAddress – if no address parameter is provided, and no last paired device exists in memory.

ConnectServiceFailed – either SPP serice or UUID service on remote device was not found, or not available for connections.

ConnectReject NotConnectable – remote device rejects the incoming attempt because it is not currently connectable.

ConnectReject ByRemote – security or other connection rejection by the remote device.

18.13 AT Connectable

The Connectable command will enable or disable the connectability of this device, which only affects incoming connection requests. This feature is saved in memory, and can be viewed using the AT ShowSettings command.

▲ kcSerial 2.2 and kcSerial 2.4 compatibility: The previous command AT+ZV UpdatePageScan has now been split into this new AT Connectable command and the AT ConnectScan command.

Command `AT Connectable <e/d>`

<e/d> `e=Enabled, d=Disabled`

Example
```
AT Connectable E
-> Connectable Enabled
```

18.14 AT ConnectDun

The ConnectDun command is used to initiate a connection with the specified device. The remote device address must be specified.

▲ kcSerial 2.2 and kcSerial 2.4 compatibility: This was previously the AT+ZV DunConnect command, but is still recognized.

Command `AT ConnectDun <btaddr>`

<btaddr> `The 12 digit Bluetooth address of the remote device`

Example
```
AT ConnectDun 0123456789AB
-> ConnectionUp
-> [BypassMode]
```

18.15 AT ConnectIOS

The ConnectIOS command is used to initiate a connection with the Apple iPhone/iPad/iPod “Wireless iAP” RfComm Bluetooth service UUID: 00000000-deca-fade-deca-deafdecacafe.

Note: an Apple authentication chip must be deployed in order to transmit or receive data. This requires an MFi license with Apple.

Command `AT ConnectIOS <btaddr>`

<btaddr> `The 12 digit Bluetooth address of the remote device`

Example
```
AT ConnectIOS
-> ConnectionUp
-> [BypassMode]
```

18.16 AT ConnectScan

The ConnectScan command is used to set the window and interval parameters allocated to scanning of incoming connection requests from remote devices. Power consumption increases when the window scan time is increased, and when the intervals between scans are decreased.

The Bluetooth default is scanning for connection requests within a window of 11.25ms (18 slots) at 1.28s intervals (2048 slots). The KC Wirefree default scanning for connection requests is twice as long, and twice as often. The parameters are the numbers of Bluetooth time slots which is 625µs per slot.

▲ kcSerial 2.2 and kcSerial 2.4 compatibility: The previous command AT+ZV UpdatePageScan has now been split into this AT ConnectScan command and the new AT Connectable command.

Command AT ConnectScan <window> <interval>

<window> 18-4096 slots (KC default 36, BT default 18)
<interval> 18-4096 slots (KC default 1024, BT default 2048)

Example AT ConnectScan 72 2048
-> ConnectScan 72 2048

18.17 AT DebugMode

The Debug command enables internal firmware debugging messages to display during operation. This can be useful for getting additional information when encountering problems. This feature is saved in memory, and can be viewed using the AT ShowSettings command.

Command AT DebugMode <e/d>

<e/d> e=Enabled, d=Disabled

Example AT DebugMode E
-> DebugMode Enabled

18.18 AT DeepSleep

The DeepSleep command enables the low power chip sleep mode for operational power savings. Please refer to the power savings section in this User Guide for additional notes regarding Deep Sleep mode. This feature is saved in memory, and can be viewed using the AT ShowSettings command.

Command AT DeepSleep <e/d>

<e/d> e=Enabled, d=Disabled

Example AT DeepSleep E
-> DeepSleep Enabled

18.19 AT Disconnect

The Disconnect command is used to terminate a connection with the remote device. It may be necessary to issue the Escape Sequence and switch to CommandMode to issue this command.

▲ kcSerial 2.2 and kcSerial 2.4 compatibility: This was previously the AT+ZV SppDisconnect command, but is still recognized.

Command AT Disconnect

Example AT Disconnect
-> Disconnect

18.20 AT DisconnectDun

The DisconnectDun command is used to terminate a connection with the remote device. It may be necessary to switch to CommandMode, or issue an EscapeCmd prior to issuing this command.

▲ kcSerial 2.2 and kcSerial 2.4 compatibility: This was previously the AT+ZV DunDisconnect command, but is still recognized.

Command `AT DisconnectDun`

Example
```
~~~~~1
-> [CommandMode]
AT DisconnectDun
-> DunConnectionClosed
```

Example
```
~~~~~2AT DisconnectDun
-> DunConnectionClosed
-> [CommandMode]
```

18.21 AT Discoverable

The Discoverable command will enable or disable the discoverability of this device. The current setting is displayed in the AT ShowSettings command. This feature is saved in memory, and can be viewed using the AT ShowSettings command.

▲ kcSerial 2.2 and kcSerial 2.4 compatibility: This was previously AT+ZV UpdateInquiryScan command, and has now been split into this new AT Discoverable command and the AT InquiryScan command.

Command `AT Discoverable <e/d>`

<e/d> `e=Enabled, d=Disabled`

Example
```
AT Discoverable E
-> Discoverable Enabled
```

18.22 AT DiscoverSvc

The DiscoverSvc command will display the Bluetooth profile service name, uuid, and channel number for each available service on the specified remote device. The additional uuid and channel number information was added to Build 029 and later. With these additions, AT DiscoverChan command was eliminated.

Command `AT DiscoverSvc <btaddr>`

<btaddr> `The 12 digit Bluetooth address of the remote device`

Example
```
AT DiscoverSvc 001060521193
-> [DiscoverServices]
-> Device[001060521193] Name[BTDELL]
-> Chan[-] Uuid[110A] Service[Bluetooth Audio Source]
-> Chan[1] Uuid[1101] Service[Bluetooth Serial Port(COM4)]
-> Chan[2] Uuid[1101] Service[Bluetooth Serial Port(COM5)]
-> Chan[-] Uuid[110C] Service[A/V Remote Control Target]
-> Chan[-] Uuid[110B] Service[Bluetooth Audio Sink]
```

```
-> Chan[6] Uuid[1123] Service[Bpp Quick Print Service]
-> Chan[9] Uuid[1101] Service[Bluetooth Serial Port (COM8)]
-> Chan[10] Uuid[111F] Service[Hands-free audio gateway]
-> Chan[11] Uuid[1112] Service[Headset audio gateway]
-> Chan[12] Uuid[111E] Service[Hands-free unit]
-> Chan[13] Uuid[1108] Service[Headset unit]
-> Chan[3] Uuid[1105] Service[Bluetooth Object Push]
-> Chan[4] Uuid[1106] Service[Bluetooth File Transfer]
-> Chan[5] Uuid[111B] Service[Generic Imaging Push service]
-> Chan[8] Uuid[1120] Service[BPP RefObject Service]
-> [EndDiscoverServices]
```

18.23 AT Discovery

The Discovery command is used to initiate a device discovery, which has 2 stages. The number of devices listed is limited to 10 total. An initial message is issued when starting the discovery process.

During the first stage, the total number of devices found will be reported (devices that are currently discoverable). A message is issued indicating the total number of devices found.

Then in stage two, a connection is made to each discovered device, in order to obtain the device name, and class of device parameter. Messages are issued one at a time, for each device, after obtaining the information requested. If some devices do not respond within a time limit, their listing is not displayed, and a DiscTimeout message is displayed at the end of the list.

If a discovered device is not connectable, then no message is printed for that device.

Command `AT Discovery <cod/fast>`

<cod/fast> `Either a cod filter, or 'F' for faster discovery without device name retrieval`

Example

```
AT Discovery
-> [Discovery]
-> Found 2 Devices, Reading Info
-> Device[001060521193] CoD[10010C] Name[BTDELL] Type[Computer]
-> Device[A87E33A1DFBD] CoD[520204] Name[Nokia 2330c-2b] Type[Phone]
-> [EndDiscovery]
```

Example

```
AT Discovery F
-> [Discovery]
-> Found 2 Devices, Reading Info
-> Device[001060521193] CoD[10010C] Type[Computer]
-> Device[A87E33A1DFBD] CoD[520204] Type[Phone]
-> [EndDiscovery]
```

18.24 AT DiscoveryRssi

The DiscoveryRssi command is used to initiate a device discovery, which has 2 stages. The number of devices listed is limited to 10 total. An initial message is issued when starting the discovery process.

During the first stage, the total number of devices found will be reported (devices that are currently discoverable). A message is issued indicating the total number of devices found. Note: with the RSSI reading, devices will respond multiple times, with current RSSI reading numbers.

Then in stage two, a connection is made to each discovered device, in order to obtain the device name, and class of device parameter. Messages are issued one at a time, for each device, after obtaining the information requested. If some devices do not respond within a time limit, their listing is not displayed, and a DiscTimeout message is displayed at the end of the list.

If a discovered device is not connectable, then no message is printed for that device.

Command `AT DiscoveryRssi`

Example
```
AT DiscoveryRssi
-> [DiscoveryRssi]
-> Found 10 Devices, Reading Info
-> Device[001060521193] CoD[10010C] RSSI[-34] Name[BTDELL] Type[Computer]
-> Device[001060521193] CoD[10010C] RSSI[-34] Name[BTDELL] Type[Computer]
-> Device[A87E33A1DFBD] CoD[520204] RSSI[-85] Name[Nokia 2330c-2b] Type[Phone]
-> Device[A87E33A1DFBD] CoD[520204] RSSI[-85] Name[Nokia 2330c-2b] Type[Phone]
-> Device[A87E33A1DFBD] CoD[520204] RSSI[-85] Name[Nokia 2330c-2b] Type[Phone]
-> Device[001060521193] CoD[10010C] RSSI[-34] Name[BTDELL] Type[Computer]
-> Device[001060521193] CoD[10010C] RSSI[-34] Name[BTDELL] Type[Computer]
-> Device[001060521193] CoD[10010C] RSSI[-34] Name[BTDELL] Type[Computer]
-> Device[001060521193] CoD[10010C] RSSI[-34] Name[BTDELL] Type[Computer]
-> Device[001060521193] CoD[10010C] RSSI[-34] Name[BTDELL] Type[Computer]
-> [EndDiscovery]
```

18.25 AT EscapeCommand

The EscapeCommand command is used to enable/disable switching to CommandMode or executing EscapeCmds. When this feature is enabled, the data stream is monitored for special EscapeMode and EscapeCmd character sequences received from the local UART port. This does reduce maximum throughput to around 115kbps.

If maximum throughput is desired, disable both EscapeCommand and RemoteCommand to prevent parsing the data stream for the special character sequences. When both EscapeCommand and RemoteCommand feature are disabled, the wireless data stream is connected directly to the UART, and allows throughput exceeding 320kbps. Use the InputCmdMode feature to enable external button switching into and out of CommandMode, and retain the high throughput, direct data stream to UART connection. This feature is saved in memory and can be viewed with the AT ShowSettings command.

Command `AT RemoteCommand <e/d>`

<e/d> `e=Enabled, d=Disabled`

Example
```
AT RemoteCommand E
-> RemoteCommand Enabled
```

18.26 AT FactoryReset

The FactoryReset command will replace all user settings with the kcSerial 3.0 default settings, and delete all paired device information. Issuing the FactoryReset command will print the current settings using AT ShowSettings, change all settings to the factory default, print the new settings using AT ShowSettings, and then reset the device.

Command AT FactoryReset

Example
```
AT FactoryReset
-> [FactoryResetBefore]
->
-> [Settings]
```

... all previous settings displayed here

```
-> [EndSettings]
->
-> [FactoryResetAfter]
->
-> [Settings]
```

... all new settings displayed here

```
-> [EndSettings]
->
-> [EndFactoryReset]
-> [Reboot]

-> kcSerial 3.0
-> by KC Wirefree
-> [CommandMode]
```

18.27 AT HciMode

This command restarts the device in HciMode, which is a raw Bluetooth mode that does not have the kcSerial command interface. After restart, HciMode will begin output of a continuous beacon signal. This mode is provided for firmware upgrading, and for testing and development applications that will directly operate the device in raw mode.

Command AT HciMode

Example
```
AT HciMode
-> HciMode [Reboot]
?@A?@A?@A?@A?@A?@A...
```

18.28 AT HwFlowControl

The HwFlowControl command is used to enable or disable use of the UART hardware flow control. This feature is recommended for high baud rate applications. When enabled, the UART CTS and RTS control lines are used. The module will reboot when changed. This feature is saved in memory and can be viewed with the AT ShowSettings command.

▲ kcSerial 2.2 compatibility: This replaces the AT+ZV StreamingSerial command.

Command AT HwFlowControl <e/d>

<e/d> e=Enabled, d=Disabled

Example AT HwFlowControl E

```
-> HwFlowControl Enabled [Reboot]
```

18.29 AT InputCmdMode

The InputCmdMode command enables or disables switching between CommandMode and BypassMode via PIO pin. A HIGH signal on the assigned PIO pin will toggle between CommandMode and BypassMode. This feature is useful for providing access to CommandMode if the EscapeCommand is disabled. By disabling EscapeCommands and RemoteCommands, kcSerial provides maximum wireless data throughput by not parsing the data in search of the EscapeMode and RemoteMode character sequences. The feature is saved in memory and can be viewed with the AT PioSettings command.

Command `AT InputCmdMode <e/d> <pio>`

<e/d> `e=Enabled, d=Disabled`

<pio> `PIO assigned to this feature`

Example
```
AT InputCmdMode E 5
-> InputCmdMode Enabled pio[5]
```

18.30 AT InputConnect

The InputConnect command enables or disables the auto connect feature which will initiate a connection or drop a connection when a HIGH signal is received on the assigned PIO pin. If the AutoConnect feature has been setup, then this will be started, otherwise, the InputConnect feature operates as the AT Connect command or AT Disconnect command. This feature is useful for providing a physical connect/disconnect button. The feature is saved in memory and can be viewed with the AT PioSettings command.

Command `AT InputConnect <e/d> <pio>`

<e/d> `e=Enabled, d=Disabled`

<pio> `PIO assigned to this feature`

Example
```
AT InputConnect E 7
-> InputConnect Enabled pio[7]
```

18.31 AT InputSleepBlock

The InputSleepBlock command allows PIO control over Deep Sleep mode. A HIGH signal on the PIO will block the device from switching into sleep mode, while a LOW signal allows normal operation of sleep mode whenever possible. If this feature is enabled, then the assigned PIO pin is unavailable for other use. This assignment will fail if another feature is currently assigned to the specified PIO. Use AT PioSettings to review all features assigned to PIO pins. The current setting is displayed with the AT PioSettings command.

Command `AT InputSleepBlock <e/d> <pio>`

<e/d> `e=Enabled, d=Disabled`

<pio> `PIO assigned to this feature`

Example
```
AT InputSleepBlock E 6
-> InputSleepBlock Enabled pio[6]
```

18.32 AT InquiryScan

The InquiryScan command is used to modify the window and interval parameters used to scan for incoming discovery requests from remote devices. Power consumption increases when the window scan time is increased, and when the intervals between scans are decreased.

The Bluetooth default is scanning for discovery requests with a window of 11.25ms (18 slots) at 2.56s intervals (4096 slots). The KC Wirefree default scanning for inquiry requests is twice as long, and twice as often. The parameters are the numbers of Bluetooth time slots which is 625µs per slot. The current setting is displayed with the AT ShowSettings command.

▲ kcSerial 2.2 and v2.4 compatibility: The previous command AT+ZV UpdateInquiryScan has now been split into this AT InquiryScan command and the new AT Discoverable command.

Command `AT InquiryScan <window> <interval>`

<window> `18-4096 slots (KC default 36, BT default 18)`
<interval> `18-4096 slots (KC default 2048, BT default 4096)`

Example
```
AT InquiryScan 72 2048
-> InquiryScan 72 2048
```

18.33 AT LinkTest

The LinkTest command is used to provide link quality information between the local device and a designated remote. The number of bytes sent in the test packet, and the BitErrorRate measurement is provided.

Command `AT LinkTest <btaddr> <iterations*>`

<btaddr> `The 12 digit Bluetooth address of the remote device`
<iterations> `The number of packets to send`

Example
```
AT LinkTest 000000000001 10
-> LinkTest 1983 Bytes BER%=[0.0000] q=255
-> LinkTest 1983 Bytes BER%=[0.0250] q=245
-> LinkTest 1983 Bytes BER%=[0.0275] q=244
-> LinkTest 1983 Bytes BER%=[0.0275] q=244
-> LinkTest 1983 Bytes BER%=[0.0225] q=246
-> LinkTest 1983 Bytes BER%=[0.1800] q=214
-> LinkTest 1983 Bytes BER%=[0.1800] q=214
-> LinkTest 1983 Bytes BER%=[0.1800] q=214
-> LinkTest 1983 Bytes BER%=[0.1800] q=214
-> LinkTest 1983 Bytes BER%=[0.1800] q=214
-> [LinkTest Done]
```

18.34 AT LinkTimeout

The LinkTimeout command is the timeout setting for an unresponsive remote connection. This feature is saved in memory and can be viewed with the AT ShowSettings command.

Command `AT LinkTimeout <time>`

<time> Time in ms for a remote link timeout

Example AT LinkTimeout 5000
-> LinkTimeout [5000]ms

18.35 AT LowLatency

The LowLatency command configures a number of buffer and timer settings to provide Low Latency data transfers, or High Throughput data transfers. This feature is saved in memory and can be viewed with the AT ShowSettings command.

Command AT LowLatency <e/d>

<e/d> e=Enabled, d=Disabled

Example AT LowLatency E
-> LowLatency Enabled

18.36 AT Messages

The HostEvent command is used to enable/disable firmware notification messages. No message is sent when successfully disabling this feature. This setting is ignored for the AT ShowSettings command.

▲ kcSerial 2.2 and v2.4 compatibility: This command replaces the AT+ZV HostEvent command. Also, the response message no longer includes an end of string NULL character (byte 00) following "Enabled".

Command AT Messages <e/d>

<e/d> e=Enabled, d=Disabled

Example AT Messages E
-> Messages Enabled

Example AT Messages D
(no response)

18.37 AT Name

The Name command is used to set the name of this device reported when other Bluetooth devices perform discoveries. The default name is "kcSerial". The name will include all characters until the <CR> marker, and does not truncate spaces. The name is saved in memory.

▲ kcSerial 2.2 and v2.4 compatibility: This command replaces the AT+ZV DefaultLocalName command, and the AT+ZV LocalName command (name change without save) is no longer available.

Command AT Name <name>

<name> Up to 32 character name. Not truncated.

Example AT Name Serial Unit
-> NameOk [Serial Unit]

18.38 AT OutputActivity

The OutputActivity feature will cause the assigned PIO to blink when wirelessly transmitting or receiving data. The Output signal will go HIGH or Blink for a minimum of 200ms when a wireless data transfer occurs. The signal is not directly related to the amount of data sent or received. If disabled, no PIO is used. See Output Features section for blink descriptions. The feature is saved in memory and can be viewed with the AT PioSettings command.

▲ kcSerial 2.4 compatibility: This replaces and enhances the AT+ZV IndicatorActivity command.

Command `AT OutputActivity <e/d> <pio> <blink>`

<e/d> `e=Enabled, d=Disabled`
<pio> `pio assigned to feature`
<blink> `0=solidlow, 1=solidon, 2=slowblink, 3=fastblink, 4=blipon`

Example
```
AT OutputActivity E 6 4
-> OutputActivity Enabled pio[6] blink[FastBlink]
```

Example
```
AT OutputActivity E 8 0
-> OutputActivity Enabled pio[8] blink[Low]
```

Example
```
AT OutputActivity D
-> OutputActivity Disabled
```

18.39 AT OutputConnect

The OuputConnect command is a simple HIGH or Blinking signal that is present when the device is wirelessly connected. If disabled, no PIO is used. See Output Features section for blink descriptions. The feature is saved in memory and can be viewed with the AT PioSettings command.

▲ kcSerial 2.4 compatibility: This replaces and enhances the AT+ZV IndicatorConnect command.

Command `AT OutputConnect <e/d> <pio> <blink>`

<e/d> `e=Enabled, d=Disabled`
<pio> `pio assigned to feature`
<blink> `0=solidlow, 1=solidon, 2=slowblink, 3=fastblink, 4=blipon`

Example
```
AT OutputConnect E 2 4
-> OutputConnect Enabled pio[2] blink[BlipOn]
```

Example
```
AT OutputConnect E 3 0
-> OutputConnect Enabled pio[3] blink[Low]
```

Example
```
AT OutputConnect D
-> OutputConnect Disabled
```

18.40 AT OutputCpu

The OuputCpu command is a simple HIGH or Blinking signal that is present when the device is turned on. It is also coupled to the BatteryMonitor feature, where this OuputCpu will go LOW when the Battery voltage has dropped below minimum. If disabled, no pio is used. See Output Features section for blink descriptions. The feature is saved in memory and can be viewed with the AT PioSettings command.

▲ kcSerial 2.4 compatibility: This replaces and enhances the AT+ZV IndicatorCpu command.

Command `AT OutputCpu <e/d> <pio> <blink>`

<e/d> `e=Enabled, d=Disabled`
<pio> `pio assigned to feature`
<blink> `0=solidlow, 1=solidon, 2=slowblink, 3=fastblink, 4=blipon`

Example
```
AT OutputCpu E 6 1
-> OutputCpu Enabled pio[6] blink[SolidOn]
```

Example
```
AT OutputCpu E 7 2
-> OutputCpu Enabled pio[7] blink[SlowBlink]
```

Example
```
AT OutputCpu D
-> OutputCpu Disabled
```

18.41 AT OutputLowBatt

The OuputLowBatt command is a simple HIGH or Blinking signal that is activated when the device is in the low battery state, as specified by AT BatteryMon command. When this OuputLowBatt command enabled, the low battery text messages are not sent to the UART. If disabled, no pio is used. See Output Features section for blink descriptions. The feature is saved in memory and can be viewed with the AT PioSettings command.

Command `AT OutputLowBatt <e/d> <pio> <blink>`

<e/d> `e=Enabled, d=Disabled`
<pio> `pio assigned to feature`
<blink> `0=solidlow, 1=solidon, 2=slowblink, 3=fastblink, 4=blipon`

Example
```
AT OutputLowBatt E 6 1
-> OutputLowBatt Enabled pio[6] blink[SolidOn]
```

Example
```
AT OutputLowBatt E 7 2
-> OutputLowBatt Enabled pio[7] blink[SlowBlink]
```

Example
```
AT OutputLowBatt D
-> OutputLowBatt Disabled
```

18.42 AT Pair

The Pair command is used to initiate pairing with a specified device. The default Pincode for this device will be automatically sent if needed for pairing with a legacy Bluetooth device, or a different Pin number can be optionally specified here. This command does not establish a connection, but obtains and saves the necessary pairing information when secured connections are required by either device. Only one successful pair is usually required for any particular remote device. A Pair attempt can fail for several reasons including: disallowed bonding by the remote device, a previous bond key entry missing or deleted by either device, an incorrect pin code, a pairing procedure timeout, using automatic pairing option when mandatory authentication is enabled.

▲ kcSerial 2.2 and v2.4 compatibility: This was previously the AT+ZV Bond command.

Command `AT Pair <btaddr>`

<btaddr> `The 12 hex digit address of the Bluetooth device to pair with.`

Example
```
AT Pair 0123456789AB
-> Paired 0123456789AB
```

Example
```
AT Pair 0123456789AB 2233
-> PairFail
```

18.43 AT Pairable

The Pairable command is used to disallow pairing with new devices. It does not prevent connections from previously paired devices. This feature is saved in memory and can be viewed with the AT ShowSettings command.

▲ kcSerial 2.2 and v2.4 compatibility: This replaces the AT+ZV DisableBond and AT+ZV EnableBond command.

Command `AT Pairable <e/d>`

<e/d> `e=Enabled, d=Disabled`

Example
```
AT Pairable E
-> Pairable Enabled
```

18.44 AT PairingDelete

The PairingDelete command is used to erase all paired device entries.

▲ kcSerial 2.2 and v2.4 compatibility: This was previously the AT+ZV EraseBondTable command.

Command `AT PairingDelete`

Example
```
AT PairingDelete
-> PairingDeleted
```

18.45 AT PairingOption

Setting the pairing options allow configuration of Bluetooth 2.1 pairing behavior. The options include specifying input capability and output capability. Bluetooth 2.1 will determine which pairing method is to be used depending on the input/output capabilities of both devices attempting to pair. The kcSerial default PairingOption is 3. See the Security section for more information regarding behavior of these options. This feature is saved in memory, and can be viewed using the AT ShowSettings command.

Command `AT PairingOption <mode>`

<mode> `0=DisplayOnly, 1=Display+Keypad, 2=KeypadOnly, 3=Automatic, 4=Reject`

Example
```
AT PairingOption 2
-> PairingOption KeypadOnly
```

18.46 AT Passkey

This command will be used to send manual keypad responses when PairingOptions indicate keypad functionality. See the Security section for more information regarding pairing and keypad options.

Command `AT Passkey <value>`

<value> `Y/N, or the multi digit passkey confirmation requested`

Example
```
AT Passkey Y
-> SendYes
```

Example
```
AT Passkey 123456
-> SendPasskey [123456]
```

18.47 AT PinCode

The PinCode command allows changing the Pin code for pairing. Default Pincode for kcSerial is 1234. The Pincode is only required when a secure connection is requested by a legacy device. By default, kcSerial automatically sends this Pincode when a Pincode is requested. This feature is saved in memory, and can be viewed using the AT ShowSettings command. See Security section for more information regarding pairing.

Command `AT PinCode <new pin> <old pin>`

<new pin> `Valid Pincode is 1-15 alphanumeric characters.`

Example
```
AT PinCode 5555 1234
-> PinCodeChanged
```

18.48 AT PioConfig

The PioConfig command is used to configure one of the general Pio pins as an input or output. If a special feature currently assigned and enabled on a particular Pio pin, such as the OutputActivity indicator, then an error message will be generated when attempting to configure the Pio pin. These configurations are not saved in memory. All Pio pins are set as inputs by default, unless they have been assigned to special Output features. The current settings are displayed with the AT PioSettings command.

▲ kcSerial 2.2 and v2.4 compatibility This improves and replaces the AT+ZV GpioConfig command.

Command `AT PioConfig <pio> <direction>`

<pio> `The Pio pin to configure`
<direction> `I=Input, O=Output`

Example
```
AT PioConfig 8 O
-> PioConfig[8]=Output
```

Example
```
AT PioConfig 5 I
-> PioConfig[5]=Err [OutputActivity]
```

18.49 AT PioRead

PioRead will supply the current reading of the pin, the configuration input or output, and the name of any special Pio Feature currently using this pin. If the Pio pin is configured as a strong pull up or pull down, a + sign will print following the reading value.

▲ kcSerial 2.2 and kcSerial 2.4 compatibility: This improves and replaces the AT+ZV GpioRead command.

Command `AT PioRead <pio>`

<pio> `0-15, the Pio pin to read`

Example
```
AT PioRead 8
-> PioRead[8]=1 Input
```

Example
```
AT PioRead 7
-> PioRead[8]=1+ Output
```

Example
```
AT PioRead 5
-> PioRead[5]=0 Output [OutputActivity]
```

Example
```
AT PioRead 15
-> PioRead[15]=0 Input [NoPin]
```

18.50 AT PioSettings

The PioSettings will display each Pio 0-15 along with current setting and status information. The current reading, input/output configuration, and any assigned features are displayed. If a + sign is appended, then the input or output is set as a strong pull up or pull down. The NoPin feature means the Pio is not available externally for use on the module, although it may or may not be available internally on the Bluetooth chip.

Command AT PioSettings

Example
```
AT PioSettings
-> [PioSettings]
-> Pio[0]=0 Input
-> Pio[1]=0 Input
-> Pio[2]=0 Input [InputCmdMode]
-> Pio[3]=1+ Input
-> Pio[4]=1 Output [OutputCpu]
-> Pio[5]=0 Output [OutputActivity]
-> Pio[6]=0 Output [OutputConnect]
-> Pio[7]=0 Output
-> Pio[8]=0 Input
-> Pio[9]=0 Input
-> Pio[10]=0 Input
-> Pio[11]=0 Input
-> Pio[12]=0 Input [NoPin]
-> Pio[13]=0 Input [NoPin]
-> Pio[14]=0 Input [NoPin]
-> Pio[15]=0 Input [NoPin]
-> [EndPioSettings]
```

18.51 AT PioStatus

The PioStatus command provides readings for all Pio pins in a compact hexadecimal format. The Read parameter indicates the HIGH or LOW reading state, where the hexadecimal bit position [15-0] corresponds to the Pio pin number. The Dir parameter indicates the Input or Output direction of the Pio, with Input LOW and Output HIGH. The Avail parameter is a mask indicating the availability of the Pio on the actual module hardware. The mask also uses the hexadecimal bit position [15-0] to correspond with the Pio pin number, where LOW is unavailable, and HIGH is present.

E.g. Read[0005] indicates Pio's 2 and 0 are HIGH, and remaining Pio's are LOW.

E.g. Dir[B000] indicates Pio's 15,13,12 are set as Outputs, and the remaining Pio's are Inputs.

E.g. Avail[0FFF] indicates Pio's 15,14,13,12 are not available on the module.

Command AT PioStatus

Example
```
AT PioStatus
-> PioStatus Read[0005] Dir[B000] Avail[0FFF]
```

18.52 AT PioStrong

The PioStrong command is used to set a Pio pin pull up/down strength. A strong pull up/down consumes more power, but can be useful for overriding a weak pull up/down circuit.

Command AT PioStrong <pio> <value>

<pio> The Pio pin to set
<value> 0=Weak(default), 1=Strong

Example AT PioStrong 5 1

```
-> PioStrong[5]=1
```

Example
```
AT PioStrong 6 0
-> PioStrong[6]=0
```

18.53 AT PioWrite

The PioWrite command is used to set a Pio pin to high or low. A Pio pin may be set when configured as an input or output. If a setting or feature assigned to a particular Pio pin is enabled, such as the SignalCpu indicator, then an error message will be generated when attempting to write the Pio.

▲ kcSerial 2.2 and kcSerial 2.4 compatibility: This improves and replaces the AT+ZV GpioRead command.

Command `AT PioWrite <pio> <value>`

<pio> `The Pio pin to read`
<value> `The value to write, 0 or 1`

Example
```
AT PioWrite 5 1
-> PioWrite[5]=1
```

Example
```
AT PioWrite 14
-> PioWrite[14]=Err [NoPin]
```

18.54 AT RemoteCommand

The RemoteCommand command is used to enable/disable switching to RemoteMode or executing RemoteCmds that have been issued from a remote device. This does not prevent issuing remote commands from this device. This feature is saved in memory and can be viewed with the AT ShowSettings command.

Command `AT RemoteCommand <e/d>`

<e/d> `e=Enabled, d=Disabled`

Example
```
AT RemoteCommand E
-> RemoteCommand Enabled
```

18.55 AT Reset

The Reset command is used to reset the kcSerial 3.0 device. This command based reset is exactly the same as the hardware pin reset. When the reset command is received, the device will send the pending message, and reset with a 500ms delay.

Command `AT Reset`

Example
```
AT Reset
-> [Reset]

-> kcSerial 3.0
-> by KC Wirefree
```

```
-> [CommandMode]
```

18.56 AT RfcService

The RfcService command is used to register and start a custom RfComm broadcast service. Any connection to this service operates the same as the standard SPP service. This is useful for connecting to Bluetooth devices that may only provide RfComm services, and this service will be used as the reciprocal service required by the remote device. It is necessary to establish an RfComm service with this command, in order to use the ConnectRfc command. This service is not saved in memory.

Command `AT RfcService <uuid> <name>`

<uuid> `Any 16, 32, or 128-bit UUID. Hexadecimal parameter without dashes`
<name> `The service name`

Example
```
AT RfcService 0123456789ABCDEF0123456789ABCDEF HelloWorld
-> RfcommServiceCreated
```

18.57 AT RfPower

The RfPower command is used to adjust the default and maximum RF power settings.

The default transmit power is used for paging, inquiry, and their responses, and as the initial power for new ACL links. The maximum transmit power is only referenced when increasing the transmit power. Power output guaranteed from -25 dBm to +4 dBm.

Settings are rounded down to the next power table entry, 4dBm increaments. Also, the Default power value is set equal or greater than the Maximum value. Settings can be viewed with the AT ShowStatus command. These settings are not saved.

Command `AT RfPower <default> <max>`

<default> `Integer as dBm. Rounded down to 4dB increments. Cannot be higher than max.`
<max> `Integer as dBm. Rounded down to 4dB increments.`

Example
```
AT RfPower -12 0
-> RfPower Default[-12 dB] Maximum[0 dB]
```

18.58 AT RoleSwitch

The RoleSwitch command will switch the master (device A) and slave (device B) roles between connected devices. This can be useful for latency and power considerations, as the master device uses less power maintaining an active link than the slave device, and also has more consistent latency for data transmissions. With SPP connections, whichever device initiates the connection, is typically designated to be the master device. This command can alter that designation. Also see AT ShowStatus for the current device role.

Command `AT RoleSwitch`

Example
```
AT RoleSwitch
-> Role master
```

Example AT RoleSwitch
-> ErrNotConnected

18.59 AT Rssi

The Rssi command returns the current RSSI reading from a current connection. The reading is between -127 and 128. A zero reading means the range is within the Golden Range, and is good. A negative reading indicates remote device is too far, and a positive reading indicates too close. Note: Rssi is not a proper range indicator, as many Bluetooth devices adjust power levels.

Command AT Rssi

Example AT Rssi
-> RSSI: 0

18.60 AT Security

The Security command is used to enable security of the local device. Bluetooth v2.1 security mode 4 is used (this is not the same as the security level). Enabling security requires incoming and outgoing connections to be properly bonded, and disabling security allows connections without bonding. Security is disabled by default. The current settings are displayed with the AT ShowSettings command. For additional information, please refer to the security usage section in this User Guide.

▲ kcSerial 2.2 compatibility: The old parameters of None or Link are recognized for backwards compatibility, and set security level 1 or 2 respectively. The kcSerial 2.2 firmware was compliant with Bluetooth v1.2, however security in Bluetooth v2.1 is different.

▲ kcSerial 2.4 compatibility: The old parameters of E or D are recognized for backwards compatibility, and set security level 1 or 2 respectively.

Command AT Security <level>

<level> 0=None, 1=Not Enforced, 2=Standard, 3=Authenticated Only

Example AT Security 1
-> SecurityLevel 1

Example AT Security None
-> SecurityLevel 1

18.61 AT SecurityAuth

The SecurityAuth command is used to enable a mandatory authenticated pairing requirement. Note: this prevents the automatic pairing mode from pairing, since this is not an authenticated method. Please see the Security section for additional information.

Command AT SecurityAuth <e/d>

<e/d> e=Enabled, d=Disabled

Example AT SecurityAuth E
-> SecurityAuth Enabled

18.62 AT ShowSettings

This command shows the current configuration of the device settings. Also see AT ShowStatus and AT PioSettings.

Command `AT ShowSettings`

Example
```
AT ShowSettings
-> [Settings]
-> Name "kcSerial"
-> D AutoConnect 000000000000 0 0
-> D BatteryMon
-> E Connectable
-> D DebugMode
-> D DeepSleep
-> E Discoverable (Override)
-> E EscapeCommand
-> D HwFlowControl
-> E LowLatency
-> E Messages
-> E Pairable
-> E RemoteCommand
-> D SecurityAuth
-> E Sniff
-> E SppService
-> D ZvMode
-> ClassOfDevice 001F00
-> ConnectScan 36 1024
-> InquiryScan 36 2048
-> LinkTimeout 5000
-> PairingOption Automatic
-> PinCode 1234
-> PrevConnect 00043E3A4E40 "KCWirefreeDevice"
-> SecurityLevel 1
-> SniffSettings 0: Active 0 0 0 0 1
-> SniffSettings 1: Sniff 32 160 2 8 2
-> SniffSettings 2: Sniff 160 640 1 16 0
-> Uart 115200-8-e-2
-> [EndSettings]
```

18.63 AT ShowStatus

This command shows current device status settings. Local and Remote device information is shown when currently connected.

Command `AT ShowStatus`

Example (Connected)
```
AT ShowStatus
-> [ShowStatus]
-> State: Connected
-> Local: Master 000000000001 [kcSerial]
-> Remote: Slave 000000000002 [BTDell]
```

```
-> RFPowerDefault: +4 dB
-> RFPowerMaximum: +4 dB
-> Temp: 27 C
-> Ram: 8 Slots
-> BootMode: 2
-> [EndShowStatus]
```

18.64 AT Sniff

The Sniff feature allows for low power operation of active wireless links. Sniff is highly recommended, and enabled by default. Sniff mode will increase the latency, and decrease the throughput for data communications. This setting is saved, and can be viewed with AT ShowSettings. Use AT SniffSettings to configure sniff parameters.

Command `AT Sniff <e/d>`

<e/d> `e=Enabled, d=Disabled`

Example
```
AT Sniff E
-> Sniff Enabled
```

18.65 AT SniffSettings

This command will configure the sniff mode parameters. Sniff mode can increase the latency, and decrease the throughput for data communications. The current settings are displayed with the AT ShowSettings command. The min and max parameters are the numbers of Bluetooth time slots which are 0.625ms per slot. The time parameter is the duration in seconds before changing to the next table entry, unless set to 0 which means this mode is infinite duration. RF activity will cause the sniff mode to restart at the 0 table entry, which is usually set to active mode. These settings are saved in memory.

Min interval - minimum number of slots to sleep
Max interval - maximum number of slots to sleep
Attempts - how many slots the slave shall listen
Timeout - how many additional slots the slave shall listen
Time - the duration (in seconds) to remain in the current table level, 0 is infinite

Default Sniff settings:

MODE	MIN	MAX	AT	TO	T
Active	0	0	0	0	2
Sniff	32	200	1	16	30
Sniff	160	640	1	16	0

Command `AT SniffSettings <line> <mode> <min> <max> <attempts> <timeout> <time>`

<line> `Sniff table entry 0-2`
<mode> `0=Active, 1=Sniff`
<min> `Minimum slots`
<max> `Maximum slots`
<attempts> `Number of attempts`
<timeout> `Timeout slots`
<duration> `Duration in sec for current mode, 0=infinite`

Example `AT SniffSettings 0 0 0 0 0 0 2`
`-> SniffSettings line 0: Active min[0] max[0] attempts[0] timeout[0] time[2]`

Example `AT SniffSettings 1 1 32 200 1 16 30`
`-> SniffSettings line 1: Sniff min[32] max[200] attempts[1] timeout[16] time[30]`

Example `AT SniffSettings 2 1 160 640 1 16 0`
`-> SniffSettings line 2: Sniff min[160] max[640] attempts[1] timeout[16] time[0]`

18.66 AT SniffSubrate

The SniffSubrate is a Bluetooth v2.1 only feature that allows for coordinated sniff operation between devices. This can help provide extended sniff modes, and lower power connections when idle. Maximum remote latency, minimum remote timeout, and minimum local timeout values can be specified for a current connection, suitable for experimentation. This setting is not saved, and can only be issued for a current, live connection.

The remote maximum should be set to twice the value of the maximum sniff setting. The parameters are the numbers of Bluetooth time slots which are 0.625ms.

Max remote latency - The maximum time the remote device need not be present when subrating (time slots).
Min remote timeout - The minimum time the remote device should stay in sniff before entering subrating mode (time slots).
Min local timeout - The minimum time the local device should stay in sniff before entering subrating mode (time slots).

Command `AT SniffSubrate <e/d> <maxremtime> <minremtime> <minloctime>`

<e/d> `e=Enabled, d=Disabled`
<rem max> `Maximum remote latency slots`
<rem min> `Minimum remote timeout slots`
<loc min> `Minimum local timeout slots`

Example `AT SniffSubrate E 3200 2 2`
`-> SniffSubrate Enabled [3200] [2] [2]`

18.67 AT SppService

The SppService feature allows the device to operate without the defaut SPP Service. This is useful for operating a custom RfComm service device only. This setting is saved, and can be viewed with AT ShowSettings. Use AT SniffSettings to configure sniff parameters.

Command `AT SppService <e/d>`

<e/d> `e=Enabled, d=Disabled`

Example `AT SppService E`
`-> SppService Enabled`

18.68 AT Timer

A basic timer command infrastructure intended for customized firmware development. This timer will output the timer mark on the set schedule. While the internal timer is precise, the output is subject to some priority scheduling variability. The outputs are transmitted wirelessly if started from the remote device using RemoteCommand Mode.

Command `AT Timer <time> <unit>`

<time> `Decimal time value, 0=stop`
<unit> `i=milliseconds, s=seconds, m=minutes, h=hours`

Example
```
AT Timer 1 s
-> [Timer]
-> Timer[0ms]
-> Timer[1001ms]
-> Timer[2002ms]
```

Example
```
AT Timer 0
-> [EndTimer]
```

18.69 AT TimerAio

A basic timer command infrastructure intended for customized firmware development. This timer will output the Aio 1 reading on the set schedule. While the internal timer is precise, the output is subject to some priority scheduling variability. The outputs are transmitted wirelessly if started from the remote device using RemoteCommand Mode.

Command `AT TimerAio <time> <unit>`

<time> `Decimal time value, 0=stop`
<unit> `i=milliseconds, s=seconds, m=minutes, h=hours`

Example
```
AT TimerAio 1 s
-> [TimerAio]
-> AioRead[1]=0mV
-> AioRead[1]=13mV
-> AioRead[1]=0mV
-> AioRead[1]=0mV
```

Example
```
AT TimerAio 0
-> [EndTimerAio]
```

18.70 AT TimerPio

A basic timer command infrastructure intended for customized firmware development. This timer will output all of the Pio readings using a hex format, where the bit position represents the high/low status of the corresponding Pio. Such as bit 15 is status of Pio 15, and bit 0 is the status of Pio 0. While the internal timer is precise, the output is subject to some priority scheduling variability. The outputs are transmitted wirelessly if started from the remote device using RemoteCommand Mode.

Command `AT TimerPio <time> <unit>`

<time> Decimal time value, 0=stop
<unit> i=milliseconds, s=seconds, m=minutes, h=hours

Example
```
AT TimerPio 1 s
-> [TimerPio]
-> Pio[0000]
-> Pio[0000]
-> Pio[0000]
-> Pio[0000]
```

Example
```
AT TimerPio 0
-> [EndTimerPio]
```

18.71 AT Version

This command returns the current version (3.0), build number (001), and edition (Standard) of the firmware. The AT Build command provides more detailed information.

Command `AT Version`

Example
```
AT Version
-> kcSerial 3.0.001 Standard Edition
```

Example
```
AT Version
-> kcSerial 3.0.035 kcRFTest Edition
```

18.72 AT ZvMode

▲ kcSerial 2.2 and v2.4 compatibility. The ZvMode feature provides kcSerial 2.2 and v2.4 compatible output messages. This may be useful for embedded applications that are currently parsing the output messages. This feature is saved in memory, and setting can be viewed using the AT ShowSettings command.

Command `AT ZvMode <e/d>`

<e/d> e=Enabled, d=Disabled

Example
```
AT ZvMode E
-> ZvMode Enabled
```

19 User Guide Version

Version	Changes
July 27, 2010	Original release
August 20, 2010	Update some commands, and security information
November 01, 2010	Added new commands from Build 035
May 9, 2011	Added new commands from Build 043

KC Wirefree Corporation
2640 W Medtronic Way
Tempe, Arizona 85281

(602) 386-2640

www.kcwirefree.com
info@kcwirefree.com
