kcSerial 4.0 Build 901X
Firmware User Guide

October 20, 2015

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

Bluetooth®
v3.0

Contents

1 Preface

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

3 Compatibility

kcSerial 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 user interface commands and several special features described in this document are unique to kcSerial firmware.

kcSerial is Bluetooth v3.0 firmware for CSR chipsets designed to be compatible with all of our legacy firmware versions.

4 Apple iOS

kcSerial firmware supports connectivity with Apple iOS devices. An MFi© (Made For iPod) 3.0 authentication chip is required, and must be connected to Pio 3, 6, and 7 for I2C communication with the kcSerial device. You must join the MFi program with Apple Inc. in order to purchase MFi authentication chips, and register your iOS application with the iTunes store. All the necessary information and capability for MFi compatibility is included within kcSerial.

5 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 AT Commands. This allows a kcSerial unit to read, write digital and analog IO pins, and transfer data under remote control from any other Bluetooth device.

DataMode - 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.

6 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. All information can be found in the firmware upgrade user guide.

7 Help Commands

Enter `AT Help` from a terminal program to view a list of all kcSerial AT Commands. Commands indicated with /Temp can be issued with the Temp suffix (ie. `AT CodTemp`) which will change the live setting temporarily, but will not save the new setting in memory. These are most useful for experimental uses, or for highly repetitive system uses to avoid flash memory burnout.

```
AT Help
-> [CommandList]
-> AT AioRead              AT AutoConnect          AT BatteryMon
-> AT BtAddr               AT Build                AT CoD/Temp
-> AT ConfigRawBaud/Temp   AT ConfigUart/Temp      AT ConnDiscForce/Temp
-> AT Connect              AT Connectable          AT ConnectIOS
-> AT ConnectScan          AT DataMode             AT DebugMode/Temp
-> AT DeepSleep/Temp       AT Dfu                  AT DiscConfig
-> AT Disconnect           AT Discoverable         AT Discovery
-> AT DiscService          AT EscapeMode/Temp      AT FactoryReset
-> AT HidHostService       AT HwFlowControl        AT IapBundleId
-> AT IapNameAcc           AT IapNameManf          AT IapNameModel
-> AT IapNameProto         AT IapService           AT IapSettings
-> AT IapVersion           AT InputCmdMode         AT InputConnect
-> AT InputDiscoverable    AT InputSleepBlock      AT InquiryScan
-> AT LinkStatus           AT LinkTest             AT LinkTimeout/Temp
-> AT LowLatency/Temp      AT Messages/Temp        AT MpSetLink
-> AT MultiPoint           AT Name/Temp            AT NegotiateSync
-> AT OutputActivity       AT OutputConnect        AT OutputCpu
-> AT OutputDiscoverable   AT OutputLowBatt        AT PacketSize/Temp
-> AT Pair                 AT Pairable             AT PairingDelete
-> AT PairingOption/Temp   AT Passkey              AT PinCode
-> AT PioConfig            AT PioRead              AT PioSettings
-> AT PioStatus            AT PioWrite             AT Radio
-> AT RemoteMode/Temp      AT Reset                AT RfcService
-> AT RfPower/Temp         AT RoleSwitch           AT Rssi
-> AT ScoConfig            AT Security/Temp        AT SecurityAuth/Temp
-> AT ShowLinks            AT ShowSettings         AT ShowStatus
-> AT Sniff/Temp           AT SniffSettings        AT SniffSubrate
-> AT SppName              AT SppService/Temp      AT Timer
-> AT Version
->
-> EscapeSeq ~~~~~1
-> EscapeCmd ~~~~~2
-> RemoteSeq ~~~~~3
-> RemoteCmd ~~~~~4
->
-> Specific command help: AT <command> ?
-> See kcSerial UserGuide www.kcwirefree.com
-> [EndCommandList]
```

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
```

7.1 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*>
```

8 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 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 ConnDiscForce	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.
AT ConnectIos	Connect to an iOS device.
AT ConnectScan	Configures incoming connection scanning settings. Saved in memory.
AT Data	Switches from CommandMode to DataMode, when connected.
AT DebugMode	Enables internal debugging messages to assist with operational problems. Saved in memory.
AT DeepSleep	Enables Deep Sleep. Saved in memory.
AT Dfu	Restarts device in DFU mode without kcSerial. Not saved.
AT DiscConfig	Configures the discovery mode and output information. Saved in memory.
AT Disconnect	Disconnect an existing Spp connection.
AT Discoverable	Enables responses to incoming inquiry requests.
AT Discovery	Search for Bluetooth devices.
AT DiscServices	Discover Bluetooth profile services on a specified remote device.
AT EscapeMode	Enables local device escape into CommandMode from DataMode. Saved in memory.
AT FactoryReset	Restores all factory default settings in memory, and resets the device.
AT HwFlowControl	Enables hardware flow control. Saved in memory.
AT InputCmdMode	Configures PIO switching between CommandMode and DataMode. Saved in memory.
AT InputConnect	Configures PIO for connecting and disconnecting. Saved in memory.
AT InputDiscoverable	Configures PIO for initiating discoverable mode. 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 IosBundleId	Set the Apple Bundle ID for the iOS application to be connected with. Saved in memory.
AT IosNameApp	Set the application name of this iOS accessory. Saved in memory.
AT IosNameDevice	Set the device name of this iOS accessory. Saved in memory.
AT IosNameManf	Set the manufacturers name of this iOS accessory. Saved in memory.
AT IosNameModel	Set the model name of this iOS accessory. Saved in memory.
AT IosService	Enable the Bluetooth service for incoming iOS connections. Saved in memory.
AT IosSettings	Prints all the Ios name settings.
AT IosVersion	Set the hardware and firmware version numbers of this iOS accessory. Saved in memory.
AT LinkStatus	Prints information relating to a current wireless link.
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 OutputDiscoverable	Configures PIO as Discoverble mode indicator. Can assign solid or blink modes. Saved in memory.
AT OutputLowBatt	Configures PIO as Low Battery indicator. Can assign solid or blink modes. Saved in memory.
AT PacketSize	Configures the number of bytes per wireless packet.
AT Pair	Executes a pairing attempt with a remote device. Pairing info saved in memory.
AT Pairable	Enables incoming pairing requests.

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, and weak/strong pullup of a PIO pin. Not saved.
AT PioRead	Read a PIO pin.
AT PioSettings	Prints all PIO pins with state, status of in/out and weak/strong, & feature assignment.
AT PioStatus	Prints 16bit masks indicating state, i/o status, weak/strong status, hw availability.
AT PioWrite	Set a high/low value for a Pio pin. Not saved.
AT Radio	Enables the Bluetooth radio. Saved in memory.
AT RemoteMode	Enables remote device escape into RemoteMode from DataMode. Saved in memory.
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 SppName	Sets the name of the SPP service. Saved in memory.
AT SppService	Enables the SPP profile. Save in memory.
AT Timer	Starts a general timer function.
AT Version	Prints the complete firmware version information, including any special edition description.

9 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. These will all be restored, and saved, to these settings when the AT FactoryDefault command is issued.

```
AT ShowSettings
-> [Settings]
-> Device Name [kcSerial]
-> D AutoConnect 000000000000 0 0
-> D BatteryMon Low=0mV Off=0mV 0s
-> D DebugMode
-> D DeepSleep
-> E EscapeMode
-> D HwFlowControl
-> D IosService
-> E LowLatency
-> E Messages
-> E Radio
-> E RemoteMode
-> D SecurityAuth
-> D Sniff
-> E SppService
-> ClassOfDevice 001F00
-> ConnDiscForce D D
-> ConnectScan 36 1024
-> DiscTimeout 0
-> DiscConfig 000000 Std 12 30 D E EEEE
-> InquiryScan 36 2048
-> LinkTimeout 5000
-> PacketSize 650
-> PairingOption Automatic
-> PinCode 1234
-> PrevConnect 000000000000
-> RfPower Default 4dB Max 4dB
-> SecurityLevel 1
-> SniffSettings 0: Disabled
-> SniffSettings 1: Disabled
-> SniffSettings 2: Disabled
-> Uart 115200-8-N-1
-> [EndSettings]
```

Device is Connectable, Discoverable, and Pairable upon startup. The following status is the default startup state for kcSerial. These are runtime variables.

```
AT ShowStatus
-> [Status]
-> E Connectable
-> E Discoverable
-> E Pairable
-> E Radio
-> Temp: 25C
-> [EndStatus]
```

10 AT Command Syntax

kcSerial 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 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.

10.1 UART and USB Interface

kcSerial 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.

10.2 Command Entry

This Firmware User Guide provides instructions for using the standard kcSerial AT Command interface via the default UART interface. AT Commands are entered as characters (bytes) via the UART interface. kcSerial will accept either <CR> or <CRLF> characters to denote the EOL (end of line). Responses will be prefixed with the -> characters, and will include <CRLF> as the EOL indicator.

```
AT Version<CRLF>         [Hex: 41 54 20 56 65 72 73 69 6F 6E 0D 0A]
-> kcSerial 4.0<CRLF>    [Hex: 2D 3E 20 6B 63 53 65 72 69 61 6C 20 33 2E 31 0D 0A]
```

10.3 Case Insensitive Commands

All kcSerial AT Commands and parameters (except certain device and service Name parameters) are converted to lower case. This User Guide presents commands using upper case characters simply for legibility.

10.4 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.

10.5 Commands in Different Modes

The AT command interface will work in both CommandMode and RemoteMode. AT commands will not work in DFU Mode and must follow an escape sequence in order to work in DataMode.

10.6 Command Help

Every AT command can be entered followed by a question mark [?] and the list of parameters will be given. When applicable, typing in an at command without the input parameters will result in the current settings displayed. Most of those settings can also be found in the AT ShowSettings and AT ShowStatus commands, among others. Remember, a list of all the commands can be given by typing the AT Help command.

11 Modes of Operation

11.1 DFU (Device Firmware Upgrade)

The highest level command interface defined by Bluetooth SIG is the Host Controller Interface Mode (HCI), which is typically used by Bluetooth stack software operating on a PC. DFU mode places the device in HCI mode with the update DFU flag. DFU 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 DFU 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 is capable of providing raw DFU level access via UART using the `AT Dfu` command. The kcSerial device will reboot into DFU using the UART interface at the default settings, 115200-7-E-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 DFU mode will operate similar to a common Bluetooth USB dongle, except that the UART interface must be utilized with a kcSerial device.

11.2 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 DataMode while wirelessly connected in order to execute commands, and a simple AT command (`AT Data`) can switch the kcSerial device back to DataMode.

11.3 DataMode

The DataMode is the default mode when wirelessly connected, allowing completely transparent data traffic between Bluetooth devices. It is only possible when connected. 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.

11.4 RemoteMode

RemoteMode is a unique and powerful remote control mode only available from KC Wirefree. With a wireless connection, the EscapeModeRem or EscapeCmdRem escape sequence can be sent to a remote kcSerial device, by any other Bluetooth device. If RemoteCmdRem is enabled, the kcSerial device parse the incoming Wireless data stream, and finding the escape sequence, will switch into RemoteMode. RemoteMode is identical to CommandMode, except that the AT commands are received wirelessly. Also, AT Command response messages are sent wirelessly to the remote device.

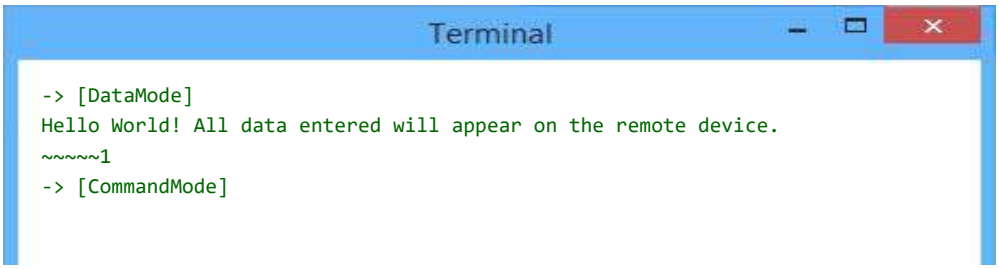
To exit RemoteMode, simply send the `AT Data` command from the remote device.

11.5 Local Escape Sequences

`AT EscapeMode`

The EscapeMode command enables/disables the local device to parse the incoming UART data, to identify either of the local escape sequences present in the data stream.

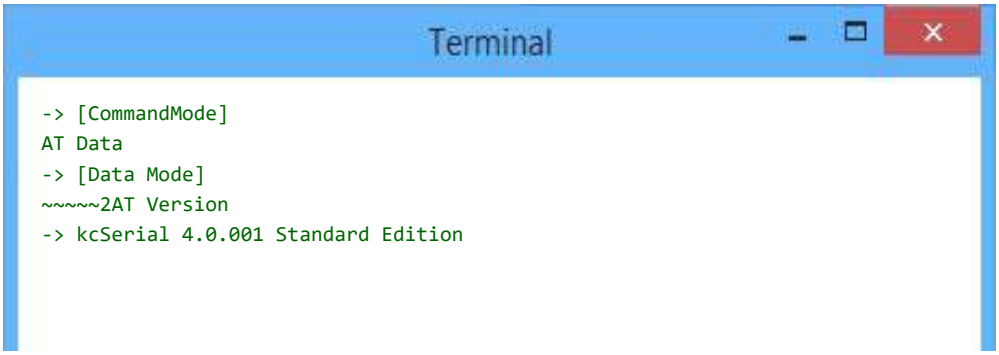
The EscapeSeq sequence `~~~~~1` will switch kcSerial into CommandMode from DataMode. Note, this escape command must be sent at one time, in a single data packet. kcSerial cannot delay valid, wireless data transmissions while waiting to process individual character keystrokes. Hyperterminal users must send a "text file" or use a key macro containing the escape sequences.



• Note: 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.

In order to switch back, the `AT Data` command can be issued. An EOL marker is not required following the EscapeSeq sequence. Any Wireless data received by the kcSerial device while switched to CommandMode, is quietly ignored, and not sent to the UART.

The EscapeCmd sequence `~~~~~2` will execute a single AT Command that is prefixed with this sequence, but not switch out of DataMode. The EOL marker is required following the AT Command.



Note: there is no space between escape sequence `~~~~~2` and the AT command.

11.6 Remote Escape Sequences

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.

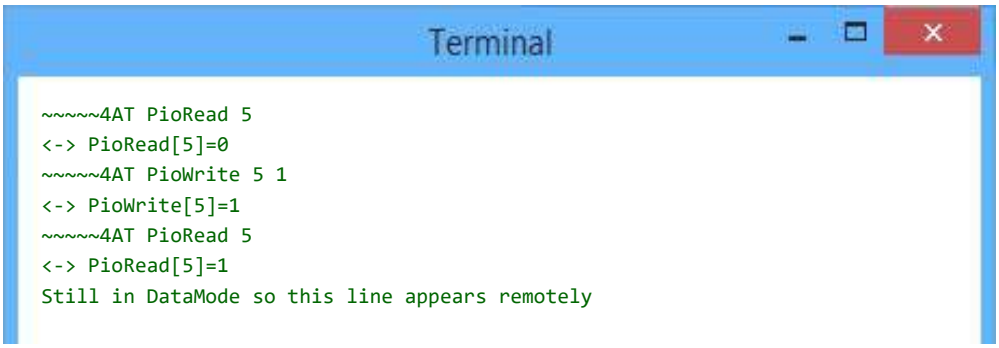
```
AT RemoteMode
```

The RemoteMode command enables/disables the local device to parse the incoming Wireless data, to identify either of the remote escape sequences present in the data stream.

The RemoteSeq sequence `~~~~~3` will switch kcSerial into RemoteMode from DataMode. In order to switch back, the `AT Data` command can be issued. An EOL marker is not required following the RemoteSeq sequence.

Note: System responses are sent wirelessly and have the <-> prefix, indicating a remote response.

The RemoteCmd sequence `~~~~~4` will execute a single AT Command that is prefixed with this sequence, but not switch out of DataMode. The EOL marker is required following the AT Command.



Note: there is no space between escape sequence ~~~~~4 and the AT command.

12 Getting Started

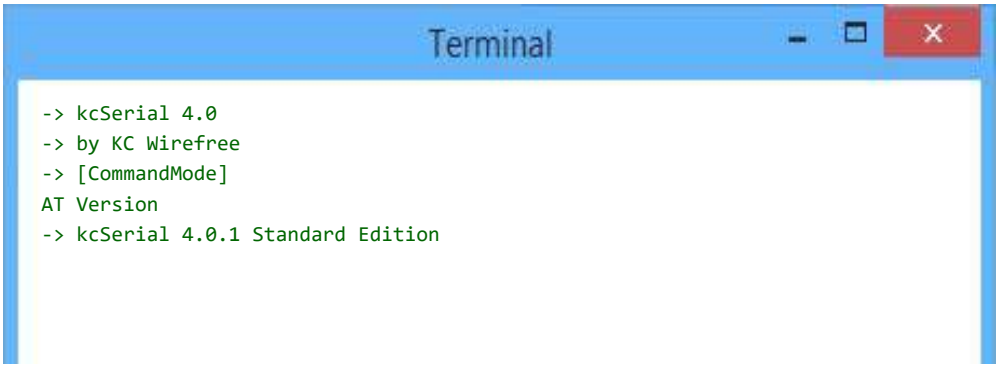
Physically connect the kcSerial 4.0 device UART pins to a PC serial port. Modules require voltage shifting from 3V3 TTL levels to match RS-232 5V voltage levels. Start a terminal program, such as HyperTerminal, and configure the port settings.

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

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

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

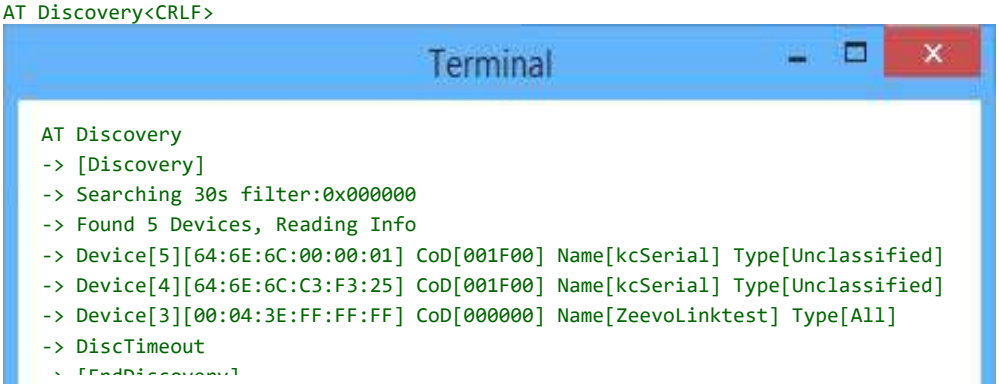
```
AT Version<CR>
```



Next, you can discover other Bluetooth devices that are currently in discoverable mode. By default, kcSerial is always Discoverable, Connectable, and Pairable when not actively connected to another device.

Enter the AT Discovery command:

```
AT Discovery<CRLF>
```



kcSerial will output several messages when initiating a search for devices, depending on the `AT DiscConfig` settings. After a complete search, the total number of devices found will be indicated. Next, kcSerial will connect to each device individually and retrieve its information including the BT address, 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, low security connection. kcSerial has security set to level 1 by default. To connect, issue this command:

```
AT Connect 00043e3a3110
```

Upon successful connection, kcSerial immediately switches into DataMode. 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 DataMode is a completely transparent transfer mode that operates as a serial cable – with one exception.

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

13 Connectable, Discoverable, Pairable

13.1 Connectable

When connectable, the device continuously wakes up to perform a wireless scan, looking for incoming connection attempts. The wakeup schedule is configurable by using the AT ConnectScan command. Connectable allows the device to receive incoming connections from devices that have previously paired. This does not affect the ability to initiate outgoing connections. kcSerial remains connectable by default, until a connection is accepted.

When connected, kcSerial will disable connectable, unless the AT ConnDiscForce setting forces the device to remain connectable while currently connected. The AT Connectable command can enable/disable the connectable status anytime. Current status can be viewed by issuing AT Connectable without parameters, or with the AT ShowStatus command.

13.2 Discoverable

When Discoverable, the device continuously wakes up to perform a wireless scan, looking for incoming inquiry requests. The wakeup schedule is configurable by using the AT InquiryScan command. When a remote device performs a discovery, and this device is currently Discoverable, it will respond to all information requests. In order to accept an incoming connection from a new device, kcSerial must be Discoverable. However, incoming connections from previously paired devices do not require Discoverable mode, since connection information has been saved during the initial pairing process.

kcSerial remains connectable by default, until a connection is accepted. Additionally, the AT DiscTimeout command can provide an automatic timeout for the Discoverable period.

When connected, kcSerial will disable Discoverable, unless the AT ConnDiscForce setting forces the device to remain Discoverable while currently connected. The AT Discoverable command can enable/disable the Discoverable status anytime. Current status can be viewed by issuing AT Discoverable without parameters, or with the AT ShowStatus command.

13.3 Pairable

When Pairable, the device accepts incoming connection requests from devices not currently stored in the paired device list (maximum 8 devices). The AT PairingDelete command will delete the entire paired device list. With Bluetooth v2.1 and higher, all connections require a successful pairing procedure in order to connect.

A pairing procedure can be initiated by a remote device with or without a connection attempt.

- If just an incoming pairing attempt is requested, then no connection will follow the pairing procedure.
- If a connection attempt is requested by a remote device that is currently stored in the local paired device list, then a connection is immediately accepted, and no pairing procedure is required.
- If a connection attempt is requested by a remote device not currently stored in the local paired device list, then the devices must successfully complete a pairing procedure prior to connection acceptance.

Please see security section of this User Guide for additional information relating to the pairing procedure, configurations, and requirements.

When the device is not Pairable, all pairing procedures will fail, which effectively blocks incoming connection attempts from devices that are not currently stored in the local paired device list.

The AT Pairable command can enable/disable the Pairable status anytime. Current status can be viewed by issuing AT Pairable without parameters, or with the AT ShowStatus command.

14 PIO Features

There are several special features that are available for Peripheral Input Output (PIO) pins. These are the AT Commands for PIO pin features and their blink settings.

14.1 InputCmdMode

A PIO input feature that causes kcSerial to switch between DataMode 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 will toggle between CommandMode and DataMode whenever this input changes from LOW to HIGH. If the device is not wirelessly connected, then it cannot switch to DataMode. This setting is saved in flash memory.

14.2 InputConnect

A PIO input feature that causes kcSerial to initiate a connection. The first choice, is to re-start any AutoConnect configuration. When AutoConnect is not configured, kcSerial 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. This setting is saved in flash memory.

14.3 InputDiscoverable

A PIO input feature that causes kcSerial to enable/disable Discoverable mode. Discoverable mode may still timeout if a discoverable timeout value is set. The InputDiscoverable feature can be disabled or assigned to any PIO pin. This setting is saved in flash memory.

14.4 InputPowerBtn

A PIO input feature that causes kcSerial to enter and exit a power off Limbo state. Limbo state disconnects kcSerial from all wireless devices before entering a low energy sleep mode. A HIGH signal to this PIO will switch the device between the Limbo state and normal operating steps. Holding this PIO with a HIGH signal for over 5 seconds will turn this PIO into a latch, where dropping the signal to LOW will act in returning the device back to the Limbo state.

14.5 InputSleepBlock

A PIO input feature that causes kcSerial 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. This setting is saved in flash memory.

14.6 OutputActivity

A simple PIO output feature that is activated when kcSerial 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. This setting is saved in flash memory.

14.7 OutputConnect

A simple PIO output feature that is activated when kcSerial 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. This setting is saved in flash memory.

14.8 OutputDiscoverable

A simple PIO output feature that is activated when kcSerial is Discoverble. The OutputDiscoverable feature can be disabled or configured to a specified PIO pin, with a choice of solid, inverted, or blinking actions. This setting is saved in flash memory.

14.9 OutputLowBatt

A simple PIO output feature that is activated when kcSerial detects a low battery voltage, as specified by the AT BatteryMon command. This setting is saved in flash memory.

14.10 OutputPwrOn

A simple PIO output feature that is activated when kcSerial is on. The OutputPwrOn 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. This setting is saved in flash memory. OutputPwrOn will go LOW in Limbo state and HIGH when leaving Limbo state.

14.11 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.

15 Auto Connect Feature

Instead of using the `AT Connect` command for connection, the auto connect 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>
```

15.1 AutoConnect

To enable the AutoConnect feature, enter the `AT AutoConnect` command with all parameters. When the AutoConnect feature is enabled, kcSerial 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
```

16 Power Saving Features

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

NOTE: This feature is disabled by default.

16.1 Deep Sleep Mode

In kcSerial, the basis for low power operation is Deep Sleep Mode, DSM. This feature temporarily halt's the chip's operation by stopping the main crystal and switching to the low power 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.

16.2 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.

16.3 InputSleepBlock

kcSerial 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.

16.4 Sniff

kcSerial 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.

17 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
```

17.1 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.

17.2 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

A random number is shown on one device, and the other device must manually enter and reply with this displayed number.

Passkey Entry

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

At least one device must enter and send a simple yes/no response to a pairing request.

Just Works

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.

17.3 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)

5 = Display+Keypad AutoHack

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. Pairing from this method is marked as Authenticated.

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. Pairing from this method is marked as Authenticated.

KeypadOnly

When using the PairingOption of KeypadOnly, keypad responses must use the `AT Passkey` command to reply with keypad entries. Pairing from this method is marked as Authenticated.

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 the pairing made by this method is marked as Not Authenticated.

Display+Keypad AutoHack

PairingOption of Display+Keypad AutoHack is designed to use with smart phones. Since both devices will indicate Display+Keyboard, each device will ask the user to confirm pairing, or confirm 6 random digits displayed on each device are the same. The AutoHack will automatically send confirmation. Pairing from this method is marked as Authenticated.

17.4 Negotiated Pairing Methods

	Device A DisplayOnly	Device A Display+Keypad	Device A KeypadOnly	Device A Automatic
Device 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]
Device 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]
Device 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]
Device 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]

17.5 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.

17.6 Authentication Required

The `AT SecurityAuth` command can be used to enable or disable 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.

17.7 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.

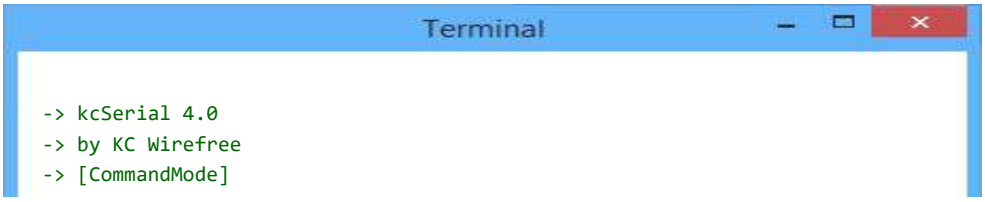
Security Level 3

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, 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.

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 -> [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 number of reconnection attempts (1000 = unlimited)
Example	AT AutoConnect D -> D
Example	AT AutoConnect E 0123456789AB 10 30 -> E 0123456789AB 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` `-> E Low=1000mV Off=800mV Period=30s`
Example	`AT BatteryMon D` `-> D`
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: v3.0 -> Hardware: KC-21 -> Firmware: kcSerial 4.0 -> Build: 035 -> Edition: Standard -> Date: Jul 25 2014 15:36:04 -> Copyright: 2014 KC Wirefree Corporation -> [EndBuildInfo]

18.7 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 -> 42E20A

18.8 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 -> 32100

18.9 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 are 115200 bps, 8 data bits, no parity, 1 stop bit, no flow control.

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	ConfigUart 115200 -> 115200-8-n-1
Example	AT ConfigUart 115200 even 2 -> 115200-8-e-2

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.

18.10 AT ConnDiscForce

The ConnDiscForce 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 ConnDiscForce <e/d> <e/d>
<e/d>	Connectable Override e=Enabled, d=Disabled
<e/d>	Discoverable Override e=Enabled, d=Disabled
Example	AT ConnDiscForce d e -> Conn[D] Disc[E]

18.11 AT Connect

The Connect command is used to initiate a connection with the specified device. Upon successful connection, kcSerial immediately switches into DataMode. 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 nor 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.

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 0123456789AB -> ConnectionUp -> [DataMode]
Example	AT Connect -> ConnectionUp -> [DataMode]
Example	AT Connect 0123456789AB 5 -> ConnectionUp -> [DataMode]
Example	AT Connect 0123456789AB 1104 -> ConnectionUp -> [DataMode]
Failure Messages	-> ErrNoBtAddress- if no address parameter is provided, and no last paired device exists in memory.
Failure Messages	-> ConnectFail [AbnormalDisconnect] - Unsuccessful due to an abnormal disconnect while establishing the RFCOMM connection. -> ConnectFail [AlreadyConnected] - The connection attempt failed because there is already a connection to that remote device on the requested RFCOMM channel. -> ConnectFail [Busy] - generic connection failure. -> ConnectFail [Failed] - generic connection failure. -> ConnectFail [NoRfcChannel] - Requested server channel not registered by this profile instance -> ConnectFail [ServiceNotFound] - Service search failed. -> ConnectFail [SlcFailed] - Service level connection establishment failed -> ConnectFail [Timeout] - Requested server channel not registered by this profile instance -> ConnectReject [NotConnectable] - remote device rejects the incoming attempt because it is not currently connectable.

18.12 AT Connectable

The Connectable command will enable/disable the connectability of this device, which only affects incoming connection requests. This feature is not saved in memory. Current status can be viewed by issuing the command without parameters, or with AT ShowStatus.

Command	AT Connectable <e/d>
<e/d>	e=Enabled, d=Disabled
Example	AT Connectable E -> E

18.13 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 -> [DataMode]

18.14 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.

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 -> 72 2048

18.15 AT Data

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

Command	AT Data
Example	AT Data -> [Data]
Example	AT Data -> ErrNotConnected

18.16 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 -> E

18.17 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 -> E

18.18 AT DFU

This command restarts the device in DFU, which is a raw Bluetooth mode that does not have the kcSerial command interface. After restart, DFU 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. DFU default Uart settings are 115200-7-Even-1.

Command	AT Dfu
Example	AT Dfu -> Dfu [Reboot] ?@A?@A?@A?@A?@A?@A?@A...

Note: Seemingly random characters A?@A.. will vary depending on Uart settings.

18.1 AT DiscConfig

The DiscConfig command configures the discovery output format. The Enabling Fast parameter will display information for each discovered device immediately, as discovered. The Enabling Full parameter will display full information for each discovered after the timeout. Full information requires connecting to each device, to obtain the device name (devices without EIR date require this secondary connection request).

Rssi and Eir modes typically produce multiple responses from each device because the RSSI reading is constantly changing. The Fast information response includes each reading along with each RSSI. The Full information response will only print device information a single time, with an averaged RSSI reading.

The Discovery process ends when either the number of device responses is received, or the timeout expires. Because RSSI and EIR modes generate multiple, continuous responses from each device, the processes usually ends sooner, when the max number of responses have been received.

Command	AT DiscConfig <cod> <mode> <num> <timeout> <fast> <full> <eeee/dddd (eir/type/name/cod)>
<cod>	sets the cod filter. 0=no filter
<mode>	std = standard discovery, rssi = include signal strength report, eir= include extended discovery data
<num>	Max number of inquiry responses (up to about 40)
<timeout>	Number of seconds to discover
<fast>	Display devices immediately as discovered.
<full>	Display full information for each discovered device.
<fields>	Information fields reported in discovery reports: e/d = eir prints eir data in Fast report e/d = type prints type of device in Fast and Full reports e/d = name prints name of device in Fast and Full reports (name only available in Fast report with EIR mode) e/d = cod prints name of device in Fast and Full reports
Example	AT DiscConfig 0 Eir 20 10 D E EEEE -> 000000 Eir 20 10 D E EEEE

Example Discovery using AT DiscConfig 0 Eir 20 10 D E EEEE.

```
AT Discovery
-> [Discovery Eir]
```

```
-> Searching 10s filter:0x000000
-> Device[00:1A:7D:DA:71:08] CoD[020104] Rssi[-42] Name[KCPC] Eir[05094B435043020A0400]
-> Device[00:1A:7D:DA:71:08] CoD[020104] Rssi[-42] Name[KCPC] Eir[05094B435043020A0400]
-> Device[64:6E:6C:C3:F3:25] CoD[001F00] Rssi[-72] Name[kcSerial] Eir[09096B6353657269616C020A000303110100]
-> Device[00:1A:7D:DA:71:08] CoD[020104] Rssi[-54] Name[KCPC] Eir[05094B435043020A0400]
-> Device[00:04:3E:FF:FF:FF] CoD[000000] Rssi[-70] Name[] Eir[]
-> Device[00:1A:7D:DA:71:08] CoD[020104] Rssi[-54] Name[KCPC] Eir[05094B435043020A0400]
-> Device[00:04:3E:FF:FF:FF] CoD[000000] Rssi[-68] Name[] Eir[]
-> Device[00:04:3E:FF:FF:FF] CoD[000000] Rssi[-68] Name[] Eir[]
-> Device[00:1A:7D:DA:71:08] CoD[020104] Rssi[-50] Name[KCPC] Eir[05094B435043020A0400]
-> Device[64:6E:6C:00:00:01] CoD[001F00] Rssi[-86] Name[kcSerial] Eir[09096B6353657269616C020A040303110100]
-> Device[00:1A:7D:DA:71:08] CoD[020104] Rssi[-59] Name[KCPC] Eir[05094B435043020A0400]
-> Device[00:1A:7D:DA:71:08] CoD[020104] Rssi[-42] Name[KCPC] Eir[05094B435043020A0400]
-> Device[64:6E:6C:C3:F3:25] CoD[001F00] Rssi[-69] Name[kcSerial] Eir[09096B6353657269616C020A000303110100]
-> Device[64:6E:6C:00:00:01] CoD[001F00] Rssi[-83] Name[kcSerial] Eir[09096B6353657269616C020A040303110100]
-> Device[00:04:3E:FF:FF:FF] CoD[000000] Rssi[-81] Name[] Eir[]
-> Device[64:6E:6C:FF:FF:FF] CoD[001F00] Rssi[-82] Name[] Eir[]
-> Device[00:04:3E:FF:FF:FF] CoD[000000] Rssi[-72] Name[] Eir[]
-> Device[64:6E:6C:00:00:01] CoD[001F00] Rssi[-81] Name[kcSerial] Eir[09096B6353657269616C020A040303110100]
-> Device[00:1A:7D:DA:71:08] CoD[020104] Rssi[-42] Name[KCPC] Eir[05094B435043020A0400]
-> Device[00:1A:7D:DA:71:08] CoD[020104] Rssi[-57] Name[KCPC] Eir[05094B435043020A0400]
->
Found 5 Devices, Reading Info
-> Device[5][00:1A:7D:DA:71:08] CoD[020104] Rssi[-48] Name[KCPC] Type[Computer]
-> Device[4][00:04:3E:FF:FF:FF] CoD[000000] Rssi[-71] Name[ZeevoLinktest] Type[All]
-> Device[3][64:6E:6C:00:00:01] CoD[001F00] Rssi[-83] Name[kcSerial] Type[Unclassified]
-> DiscTimeout
-> [EndDiscovery]
```

18.2 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.

Command	`AT Disconnect`

Example	`AT Disconnect`
	`-> Disconnect`
	`-> [CommandMode]`

18.3 AT Discoverable

The Discoverable command will enable or disable the discoverability of this device. It also lets you set the time in seconds of time to discover, like the AT DiscConfig command. The current setting is displayed in the AT ShowSettings command. This feature is not saved in memory. Current status can be viewed by issuing the command without parameters, or with AT ShowStatus.

Command	`AT Discoverable <e/d> <timeout*>`
<e/d>	`e=Enabled, d=Disabled`
<timeout*>	`Number of seconds to discover`
Example	`AT Discoverable E` `-> Disc[E] Timeout[0]`

Example	`AT Discoverable E 20`
	`-> E Timeout[20]`
	`-> Discoverable Disabled` (20seconds later)

18.4 AT Discovery

The Discovery command is used to initiate a device discovery has 2 stages. Please see AT DiscConfig for specific discovery configurations.

The number of devices listed is limited to a total of 10 by default.

An initial message is issued when starting the discovery process.

If the Fast discovery configuration is set, then devices are reported as they are discovered.

Then, a message is issued indicating the total number of (unique) devices found.

Finally, if the Full discovery configuration is set, a connection is made to each discovered device, in order to obtain the full device name. Reports 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 report is not displayed, and a DiscTimeout message is displayed at the end of the list.

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

Command	`AT Discovery`

Example	`AT Discovery`
	`-> [Discovery]`
	`-> Searching 30s filter:0x000000`
	`-> Device[00:1A:7D:DA:71:08] CoD[020104] Rssi[32767] Name[] Eir[]`
	`-> Device[00:04:3E:FF:FF:FF] CoD[000000] Rssi[32767] Name[] Eir[]`
	`-> Device[64:6E:6C:00:00:01] CoD[001F00] Rssi[32767] Name[] Eir[]`
	`-> Device[64:6E:6C:FF:FF:FF] CoD[001F00] Rssi[32767] Name[] Eir[]`
	`-> Device[64:6E:6C:C3:F3:25] CoD[001F00] Rssi[32767] Name[] Eir[]`
	`->`
	`Found 5 Devices, Reading Info`
	`-> Device[5][00:1A:7D:DA:71:08] CoD[020104] Name[KCPC] Type[Computer]`
	`-> Device[4][00:04:3E:FF:FF:FF] CoD[000000] Name[ZeevoLinktest] Type[All]`

	-> Device[3][64:6E:6C:00:00:01] CoD[001F00] Name[kcSerial] Type[Unclassified] -> DiscTimeout -> [EndDiscovery]

18.1 AT DiscServices

The DiscServices 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.

Command	AT DiscServices <btaddr>
<btaddr>	The 12 digit Bluetooth address of the remote device
Example (an iPhone6)	AT DiscServices 78:7E:61:4A:13:72 -> [DiscoverServices] -> Device[78:7E:61:4A:13:72] Name[CS6] -> Chan[2] Uuid[1132] Service[MAP MAS-iOS] -> Chan[-] Uuid[1116] Service[PAN Network Access Profile] -> Chan[1] Uuid[00000000DECAFADEDECADEAFDECACAFE] Service[Wireless iAP] -> Chan[-] Uuid[110E] Service[AVRCP Device] -> Chan[-] Uuid[110C] Service[AVRCP Device] -> Chan[-] Uuid[110A] Service[Audio Source] -> Chan[13] Uuid[112F] Service[Phonebook] -> Chan[8] Uuid[111F] Service[Handsfree Gateway] -> Chan[-] Uuid[1000] Service[-] -> [EndDiscoverServices]

18.2 AT EscapeMode

The EscapeMode command is used to enable/disable switching to CommandMode or executing Escape Commands. When this feature is enabled, the data stream is monitored for the special EscapeSeq and EscapeCmd character sequences received from the local UART port. Parsing data streams for escape sequences reduces the throughput to around 220kbps.

If maximum throughput is desired, disable both EscapeMode and RemoteMode to prevent parsing the data stream for the special character sequences. When both EscapeMode and RemoteMode feature are disabled, the wireless data stream is connected directly to the UART, and allows throughputs 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 EscapeMode <e/d>
<e/d>	e=Enabled, d=Disabled
Example	AT EscapeMode E -> E

18.3 AT FactoryReset

The FactoryReset command will replace all user settings with the kcSerial default settings, and delete all paired device information, and then reset the device. FactoryReset is also triggered by holding PIO 3 HIGH during power up.

Command	AT FactoryReset
Example	AT FactoryReset -> FactoryReset [Reboot] -> kcSerial 4.0 -> by KC Wirefree -> [CommandMode]

18.4 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.

Command	AT HwFlowControl <e/d>
<e/d>	e=Enabled, d=Disabled
Example	AT HwFlowControl E -> E [Reboot]

18.5 AT InputCmdMode

The InputCmdMode command enables or disables switching between CommandMode and DataMode via PIO pin. A HIGH signal on the assigned PIO pin will toggle between CommandMode and DataMode. 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 -> E Pio[5]

18.6 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 -> E Pio[7]

18.7 AT InputDiscoverable

The InputDiscoverable command enables or disables the Discoverable feature when a HIGH signal is received on the assigned PIO pin. Discoverable mode may still timeout if a discoverable timeout value is set. The InputDiscoverable feature operates the AT Discoverable command, toggling enable and disable when a HIGH signal is received. It can be disabled or assigned to any PIO pin. The feature is saved in memory and can be viewed with the AT PioSettings command.

Command	AT InputDiscoverable <e/d> <pio>
<e/d>	e=Enabled, d=Disabled
<pio>	PIO assigned to this feature
Example	AT InputConnect E 5 -> E Pio[5]

18.1 AT InputPwrBtn

The InputPwrBtn command allows or blocks Limbo state entrance and exit by PIO signal. When this input feature is enabled, a HIGH signal on this pin will wake the device into either a state of normal operation from Limbo state, or a state of Limbo from normal operation. If held HIGH for over 5 seconds, it will act as a latch. This means LOW will drop the device into Limbo state. The InputPwrBtn feature can be disabled or assigned to any PIO pin. The feature is saved in memory and can be viewed with the AT PioSettings command.

Command	`AT InputPwrBtn <e/d> <pio>`
<e/d>	`e=Enabled, d=Disabled`
<pio>	`PIO assigned to this feature`
Example	`AT InputPwrBtn E 6` `-> E Pio[6]`

18.2 AT InputSleepblock

The InputSleepBlock command allows or blocks DeepSleep Mode when enabled. 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. AT DeepSleep is still required to enable DeepSleep mmode to happen. The InputSleepBlock feature can be disabled or assigned to any PIO pin. The feature is saved in memory and can be viewed 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 InputConnect E 2` `-> E Pio[2]`

18.3 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.

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 -> 72 2048

18.4 AT IosBundleId

The IosBundleId command is used to set the 10 digit Apple application id number that will launch upon connection. This setting is saved to memory.

Command	AT IosBundleId <id>
<id>	The 10 digit Apple id
Example	AT IosBundleId A123456789 -> A123456789

18.5 AT IosNameApp

The IosNameApp command is used to set the Apple application name that will launch upon connection. This setting is saved to memory.

Command	AT IosNameApp <name>
<name>	App name
Example	AT IosNameApp com.kcwirefree.btdemo -> com.kcwirefree.btdemo

18.6 AT IosNameDevice

The IosNameDevice command is used to set the Apple application name that will launch upon connection. This setting is saved to memory.

Command	AT IosNameDevice <name>
<name>	Product name
Example	AT IosNameDevice KC Wirefree BTDemo -> KC Wirefree BTDemo

18.7 AT IosNameManf

The IosNameManf command is used to set the Apple application name that will launch upon connection. This setting is saved to memory.

Command	`AT IosNameManf <name>`
<name>	`Manufacturer name`
Example	`AT IosNameManf KC Wirefree` `-> KC Wirefree`

18.8 AT IosNameModel

The IosNameModel command is used to set the Apple application name that will launch upon connection. This setting is saved to memory.

Command	`AT IosNameModel <name>`
<name>	`Device model name`
Example	`AT IosNameModel Model 0` `-> Model 0`

18.9 AT IosService

The IosService command is used to make the device available for iOS connectivity. The iAP service will be created and the CoD will be changed to indicate a headset device, which are both necessary for Apple iOS to discover and connect to this device. This setting is saved to memory.

Command	`AT IosService <e/d>`
<e/d>	`Enable/disable the Ios service`
Example	`AT Ios IosVersion Service E` `-> E`

18.10 AT IosSettings

The IosSettings command prints all the Ios name settings.

Command	AT IosSettings
Example	AT IosSettings -> [IosSettings] -> IosService: Enabled -> IosNameManf: KC Wirefree -> IosNameDevice: KC Wirefree BTDemo -> IosNameModel: Model 0 -> IosNameApp: com.kcwirefree.btdemo -> IosVersion: Fw[1.0.0] Hw[1.0.0] -> IosBundleId: A123456789 -> [EndIosSettings]

18.11 AT IosVersion

The IosNameDevice command is used to set the Apple application name that will launch upon connection.

Command	AT IosVersion <fw major> <fw minor> <fw build> <hw major> <hw minor> <hw build>
<fw major>	Major firmware version number
<fw minor>	Minor firmware version number
<fw build>	Build firmware version number
<hw major>	Major hardware version number
<hw minor>	Minor hardware version number
<hw build>	Build hardware version number
Example	AT IosVersion 1 2 3 4 5 6 -> Fw[1.2.3] Hw[4.5.6]

18.12 AT LinkStatus

The LinkStatus command shows connection related information

Command	AT LinkStatus
Example	AT LinkStatus -> Connected to 646E6CC11BCA BTv4.0 KCCJ
Example	AT LinkStatus -> Not Connected

18.13 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]

18.14 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 -> [5000]ms

18.15 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 -> E

18.16 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.

Command	`AT Messages <e/d>`
<e/d>	`e=Enabled, d=Disabled`
Example	`AT Messages E` `-> Messages Enabled`
Example	`AT Messages D` (no response)

18.17 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.

Command	`AT Name <name>`
<name>	`Up to 32 character name. Not truncated.`
Example	`AT Name Serial Unit` `-> [Serial Unit]`

18.18 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.

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 -> E Pio[6] Blink[FastBlink]
Example	AT OutputActivity E 8 0 -> E Pio[8] Blink[Low]
Example	AT OutputActivity D -> D

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.

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 -> E Pio[2] Blink[BlipOn]
Example	AT OutputConnect E 3 0 -> E Pio[3] Blink[Low]
Example	AT OutputConnect D -> D

18.19 AT OutputCpu

The OuputCpu command is a simple HIGH or Blinking signal that is present when the device is turned on. 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 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 -> E Pio[6] Blink[SolidOn]
Example	AT OutputCpu E 7 2 -> E Pio[7] Blink[SlowBlink]
Example	AT OutputCpu D -> D

18.20 AT OutputDiscoverable

The OuputDiscoverable command is a simple HIGH or Blinking signal that is activated when the device is in discoverable state. 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 OutputDiscoverable <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 OutputDiscoverable E 6 1 -> E Pio[6] Blink[SolidOn]
Example	AT OutputDiscoverable E 7 2 -> E Pio[7] Blink[SlowBlink]
Example	AT OutputDiscoverable D -> D

18.21 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 -> E Pio[6] Blink[SolidOn]
Example	AT OutputLowBatt E 7 2 -> E Pio[7] Blink[SlowBlink]
Example	AT OutputLowBatt D -> D

18.22 AT PacketSize

The PacketSize command sets the number of bytes used per packet. This affects a number of internal configuration items that can improve performance with low or high data usage requirements. This feature is saved in memory and can be viewed with the AT ShowSettings command.

Command	AT PacketSize <size>
<size>	Valid size 24 to 32767. Default is 650.
Example	AT PacketSize 780 -> 780

18.23 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 an optional Pincode can be 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 pairing 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.

Command	AT Pair <btaddr> <pin*>
<btaddr>	The 12 hex digit address of the Bluetooth device to pair with.
<pin>	Optional pin 1-15 alphanumeric characters.
Example	AT Pair 0123456789AB -> Pair 0123456789AB
Example	AT Pair 0123456789AB 2233 -> Pair 0123456789AB

18.24 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 not saved in memory. Current status can be viewed by issuing the command without parameters, or with AT ShowStatus.

Command	AT Pairable <e/d>
<e/d>	e=Enabled, d=Disabled
Example	AT Pairable E -> Enabled

18.25 AT PairingDelete

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

Command	AT PairingDelete
Example	AT PairingDelete -> Ok

18.26 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 (Automatic). 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+Keys, 2=KeypadOnly, 3=Automatic, 4=Reject, 5=Display+Keypad AutoHack
Example	AT PairingOption 2 -> KeypadOnly

18.27 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 -> Yes
Example	AT Passkey 123456 -> 123456

18.28 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 (Bluetooth v2.0 or earlier). 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 <pin>
<pin>	Valid Pincode is 1-15 alphanumeric characters.
Example	AT PinCode 5555 -> 5555

18.29 AT PioConfig

The PioConfig command is used to configure one of the general Pio pins as an input or output. All Pio pins are set as inputs by default, unless they have been assigned to special Output features. 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. A Pio Feature assigned to a pio pin must be disabled prior re-configuring the pio. The pio configurations are not saved in memory. Current settings are displayed with the AT PioSettings command.

Command	AT PioConfig <pio> <i/o> <+ pull*>
<pio>	The Pio pin to configure
<i/o>	i=Input, o=Output
<+ pull>	+=StrongPull up/down
Example	AT PioConfig 8 O + -> Pio[8] = Output+
Example	AT PioConfig 8 I -> Pio[8] = Input
Example	AT PioConfig 5 I -> Pio[5] = Err[OutputActivity]

18.30 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/down, a '+' sign will print following configuration.

Command	AT PioRead <pio>
<pio>	0-15, the Pio pin to read
Example	AT PioRead 8 -> Pio[8] = 1 Input+
Example	AT PioRead 7 -> Pio [7] = 1 Output+
Example	AT PioRead 5 -> Pio[5] = 0 Output [OutputActivity]
Example	AT PioRead 15 -> Pio [15] = 0 Input [NoPin]

18.31 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 [NoPin] -> Pio 13 [NoPin] -> Pio 14 [NoPin] -> Pio 15 [NoPin] -> [EndPioSettings]

18.32 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 -> Read[0005] Dir[B000] Avail[0FFF]

18.33 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.

Command	AT PioWrite <pio> <value>
<pio>	The Pio pin to write
<value>	The value to write, 0 or 1
Example	AT PioWrite 5 1 -> Pio[5] = 1
Example	AT PioWrite 14 0 -> Pio[14] = Err[NoPin]

18.34 AT Radio

The Radio command is used to enable/disable of the Bluetooth radio. This feature is not saved in memory. Current status can be viewed by issuing the command without parameters, or with AT ShowStatus.

Command	AT Radio <e/d>
<e/d>	e=Enabled, d=Disabled
Example	AT Radio E -> E

18.35 AT RemoteMode

The RemoteMode command is used to enable/disable switching to RemoteMode or executing Remote Commands that have been issued from a remote device. This feature is saved in memory and can be viewed with the AT ShowSettings command.

Command	AT RemoteMode <e/d>
<e/d>	e=Enabled, d=Disabled
Example	AT RemoteMode E -> E

18.36 AT Reset

The Reset command is used to reset the kcSerial 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 4.0 -> by KC Wirefree -> [CommandMode]

18.37 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. Current settings can be viewed by other devices in a discover services scan.

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 -> HelloWorld

18.38 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 -> Default[-12dB] Maximum[0dB]

18.39 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`

18.40 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`

18.41 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. Changing security level resets the device. The current settings are displayed with the AT ShowSettings command. For additional information, please refer to the security usage section in this user guide.

Command	`AT Security <level>`
<level>	`0=None, 1(D,N)=Not Enforced, 2(E,L)=Standard, 3(S)=Authenticated Only`
Example	`AT Security 1` `-> 1`
Example	`AT Security e` `-> 2`

18.42 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 D` `-> D`
Example	`AT SecurityAuth E` `-> Level protocol[1] channel[1] level[1] inout[T] auth[T] legacy[F]`

18.43 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]`
	`-> Device 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 IosService`
	`-> E LowLatency`
	`-> E Messages`
	`-> E Pairable`
	`-> E Radio`
	`-> E RemoteCommand`
	`-> D SecurityAuth`
	`-> E Sniff`
	`-> E SppService`
	`-> PacketSize 650`
	`-> ClassOfDevice 240404`
	`-> ConnectScan 36 1024`
	`-> DiscConfig 000000 Std 12 30 D E EEEE`
	`-> 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 2 16 30 -> SniffSettings 2: Sniff 160 640 2 16 0 -> Uart 115200-8-N-1 -> [EndSettings]

18.44 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 -> [Status] -> E Connectable -> E Discoverable -> E Pairable -> E Radio -> Temp: 31C -> [EndStatus]

18.45 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 -> E

18.46 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 – duration the device listens each time it wakes from its timeout period. This is the listening duration. Higher durations decrease timing mismatch issues, increase reliability, and increase power consumption.

Timeout - how many additional slots available within the wakeup period to receive data. Higher duration increases throughput when required.

Time - the duration (in seconds) to remain at the current table level, 0 is infinite

Default Sniff settings:

```
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: Active 0 0 0 0 2 -> 1: Sniff 32 200 2 16 30 -> 2: Sniff 160 640 2 16 0
Example	AT SniffSettings 1 1 32 200 1 16 30 -> line 1: Sniff min[32] max[200] attempts[1] timeout[16] time[30]
Example	AT SniffSettings 2 1 160 640 1 16 0 -> line 2: Sniff min[160] max[640] attempts[1] timeout[16] time[0]

18.47 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` `-> E [3200] [2] [2]`

18.48 AT SppName

The SppName feature sets the name of the SPP service. This setting is saved.

Command	`AT SppName <name>`
<name>	`Name of the SPP service`
Example	`AT SppName MySpp` `-> SppName MySpp`

18.49 AT SppService

The SppService feature allows the device to operate without the default SPP Service. This is useful for operating a custom RfComm service device. 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 -> E

18.50 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> <unit>	Decimal time value, 0=stop i=milliseconds, s=seconds, m=minutes, h=hours
Example	AT Timer 1 s -> [StartTimer] -> Timer = 7 ms -> Timer = 1001 ms -> Timer = 2002 ms
Example	AT Timer 0 -> [EndTimer]

18.51 AT Version

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

Command	AT Version
Example	AT Version -> kcSerial 4.0 b1 Standard Edition
Example	AT Version -> kcSerial 4.0 b1 kcRFTest Edition

19 User Guide Version

Version	Changes

KC Wirefree Corporation
2640 W Medtronic Way
Tempe, Arizona 85281
(602) 386-2640
www.kcwirefree.com
info@kcwirefree.com
