EmbeddedBlue™ Wireless Serial Port

Firmware Reference Manual (ebSerial version 2.1) March 26, 2007

Table of Contents

This page intentionally left blank.

Introduction

This document describes A7 Engineering's EmbeddedBlue Wireless Serial Port firmware (ebSerial) version 2.1. This firmware will run on most EmbeddedBlue serial devices including the eb100-SER, eb500-SER, and eb501-SER. The most recent firmware release can be downloaded from the A7 Engineering website at www.a7eng.com.

Manual Conventions

Below is a list of typographical conventions used in this manual:

`Text in this font`

- `Is used to show data that is sent to the EmbeddedBlue device.`
- `Inside a gray box is used to show data that is sent from the EmbeddedBlue device.`

In the command set section of this manual

- Required parameters and placeholders appear in standard lowercase type.
- Placeholders appear in *italics*. For example, if *address* shows up in a syntax line, the actual address of the device must be entered.
- Parameter options are separated by a vertical bar |.
- Optional parameters are enclosed in brackets [ ].

Getting More Information

General information regarding the ebSerial firmware, EmbeddedBlue, and other Bluetooth products from A7 Engineering can be found on the A7 website at www.a7eng.com.

A7 Engineering provides technical support for EmbeddedBlue products through an online discussion forum at www.a7eng.com/support/forum/forum.htm. When you visit the forum you can search through previously asked questions for information or post new ones. The forum is monitored by A7 Engineering employees so that your question will be answered in a thorough and timely manner. If your question involves sensitive information you can request private support by sending an email to support@a7eng.com.

This page intentionally left blank.

The Basics

The EmbeddedBlue ebSerial firmware is designed to provide an easy to use wireless serial port capable of communicating with a wide array of standard Bluetooth devices such as cell phones, PDAs, PCs, and other ebSerial devices. It can be used as a standalone cable replacement solution or controlled from a host processor through an easy to use command set. The primary application profile that is supported is SPP, or the Serial Port Profile. This is the most popular and convenient protocol for many embedded applications of Bluetooth since it emulates a simple serial port link between devices. Once the connection is established, communications between the endpoints is the same as for a wired serial port.

The eb100-SER supports two main operating modes: EasyConnect™ mode and command mode. In EasyConnect mode the device operates as a simple cable replacement solution. In command mode there is a rich set of functions that allow the host to have programmatic control over the module. In both modes the factory default communication parameters are 9600 Baud, 8 Data Bits, 1 Stop Bit, No Parity, and No Flow Control.

EasyConnect

EasyConnect mode provides a simple cable replacement solution between either two ebSerial devices or one ebSerial device and another Bluetooth device such as a cell phone or PC. Once set up, data is transmitted between the devices automatically without the need for additional configuration or control. The wireless cable connection will be established and maintained whenever the eb100-SER is powered.

This usage is most common when you want to enable a device with wireless technology, but do not want to make any modifications to it other than connecting it to an ebSerial device. Figure 1 illustrates a cable replacement link between two ebSerial devices.

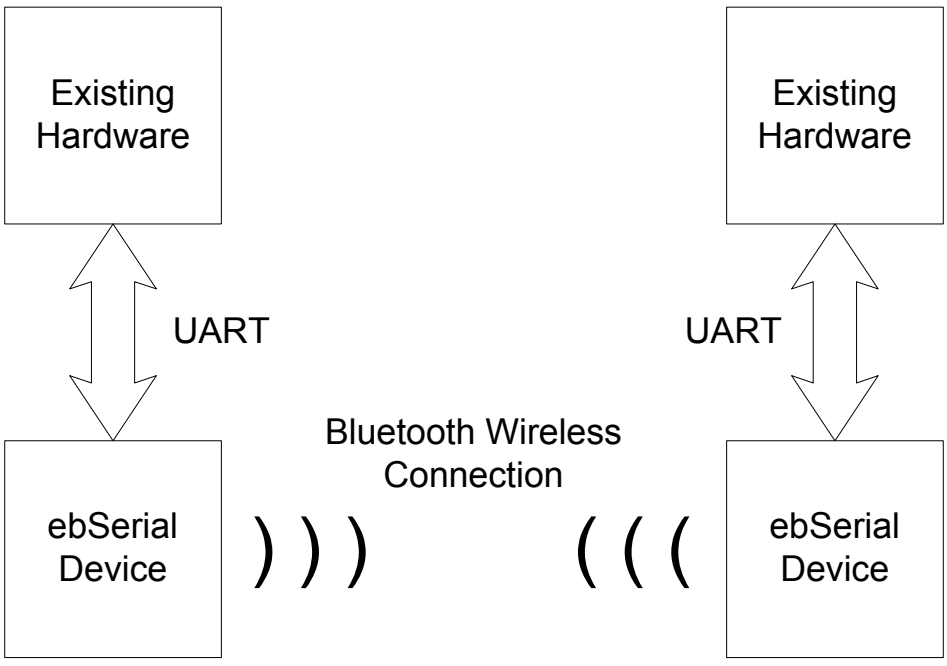


Figure 1: EasyConnect Cable Replacement Diagram

Since this scenario does not require any changes to the existing hardware, a onetime pairing procedure needs to be performed to connect the two Bluetooth devices together. Pairing two EasyConnect devices together is easy to do and is illustrated here using eb501-SER adapters as pictured in Figure 2 below.

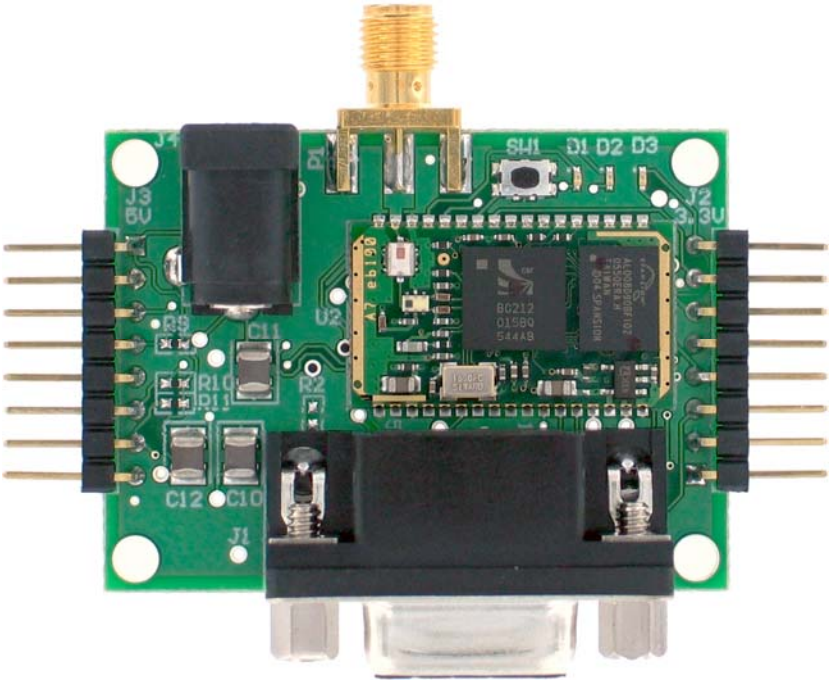


Figure 2: eb501-SER adapter

By default ebSerial devices are configured to be controlled through the serial command set and must go through the EasyConnect pairing procedure to be used as a wireless cable replacement solution. To configure an eb501-SER device for EasyConnect mode, follow these simple steps.

1. On one of the eb501-SER adapters hold down button SW1 and apply power.
2. When the LED turns on, release the button. The LED will remain on until setup is complete.
3. On the second eb501-SER adapter hold down button SW1 and apply power.
4. When the LED turns on, release the button. The LED will remain on until setup is complete.

When the LEDs on both adapters begin to blink, the paring procedure is complete. They are now logically linked to each other and will form a wireless serial cable whenever they are both powered on. When in this mode, the serial command set is disabled. You can tell if a device is configured for EasyConnect mode by watching the IND_LED line (labeled D3 on the eb501-SER) when you apply power. It will blink once if the device is in serial command mode and twice if it is in EasyConnect mode.

EasyConnect is available on the eb501-SER and eb100-SER and can be embedded into your design by using the EASY_CON and IND_LED lines as demonstrated in the eb100-SER datasheet.

Command Mode

In EasyConnect mode the ebSerial device automatically establishes a wireless connection and passes data just like a wired serial cable. If you require more control than this, the ebSerial firmware also provides you access to the full power of Bluetooth through an easy to use set of human readable serial commands. This command set effectively removes the complexity of working with Bluetooth while providing you access to the functionality that you need. Since these commands are human readable, they can easily be issued from an application on your micro or from a terminal application like HyperTerminal on the PC. Interacting with the command set is similar to using a shell environment like MS-DOS or Linux.

Devices running ebSerial firmware enter command mode by default so all that you need to do to get access is supply power and connect to the UART transmit and receive lines. The factory default communication parameters are 9600 Baud, 8 Data Bits, 1 Stop Bit, No Parity, and No Flow Control. If a device has been configured for EasyConnect mode you will need to perform a factory reset to return it to command mode. Simply hold down the EasyConnect button, apply power, and wait for approximately five seconds for the indicator LED to turn off.

Command mode provides the host with programmatic control over the module and its configuration. There are a number of commands that can be sent to change the baud rate, locate other devices that are in range, check the firmware version, etc. All commands are sent using visible ASCII characters (123 is 3 bytes “123”). Upon the successful transmission of a command, the ACK string will be returned. If there is a problem in the syntax of the transmission then a NAK string is returned. After either the ACK or NAK, a carriage-return <CR> character is returned. When a prompt (<CR> followed by a ‘>’) is returned, it means that the eb100-SER radio is in the idle state and is waiting for another command. White space is used to separate arguments of the command and a carriage-return <CR> (ASCII 0x0D) is used to mark the end of the command.

Sending and Receiving Data

Once connected to another Bluetooth device, the ebSerial firmware automatically begins to pass data wirelessly. All data transmitted to the UART will be sent to the remote device. Therefore, NO further commands can be issued until the radio is disconnected or switched back to command mode by use of the BREAK control line or the soft break command.

The connection status line of the EmbeddedBlue device can be monitored to determine if there is an active connection. The status line is active low, in other words, it will read 0V when there is an active Bluetooth connection.

This page intentionally left blank.

Command Set

The ebSerial command set operates like a command shell and is comprised of visible ASCII characters. Therefore, a command can be issued from a terminal application, such as HyperTerminal, or directly from a custom application written in a programming language such as assembly, C, Java, or Basic. From a microcontroller application, these commands are issued directly to the UART connected to the ebSerial device.

Command Basics

Commands may only be sent to the ebSerial device when it is in command mode. White spaces are used to separate parameters of the command and a carriage-return (ASCII 0x0D) is used to mark the end of the command. Upon receipt of a command the firmware begins to parse the parameters. If the syntax of the command is correct an ACK string will be returned, specifically the three bytes 'A', 'C', 'K'; otherwise, a NAK string is returned. Following the ACK or NAK string is a carriage-return character. If an error occurs while processing the acknowledged command an error string is returned followed by a carriage-return followed by the prompt (>) character. If the command executed successfully the device will issue the prompt (>) character. If the command executed successfully the module will issue the prompt (>) character. A full description of the possible error codes is available in the Error Codes section.

Note:

In the following examples, text inside of a gray box is used to show data that is sent from the ebSerial device.

The following example shows the basic structure of a command. A prompt (>) is issued by the ebSerial device. A command followed by a carriage-return is sent to the device. The device responds with either an ACK or NAK string followed by a carriage-return. If an error occurs, the device responds with an Err string followed by a space followed by an ASCII string numeric value followed by a carriage-return. A prompt (>) is then issued by the device.

```
>command<CR>
ACK | NAK<CR>
Err number<CR>
>
```

Set commands are always persisted to internal flash memory. This ensures that all settings are maintained across power cycles. For example, you only need to issue the following command once because the device name will be persisted to flash memory and maintained across power cycles:

```
>set name JoesBluetoothRadio<CR>
ACK<CR>
>
```

The EmbeddedBlue device name will now remain "*JoesBluetoothRadio*" until a user changes the name again or a "reset factory" command is issued.

Connect

The *connect* command establishes a connection to another Bluetooth device. The *connect* command may be canceled before a connection is established by issuing a carriage-return to the EmbeddedBlue device. It can take up to four seconds to cancel a connection request. Once the *connect* command has been successfully canceled, a prompt (>) is issued by the EmbeddedBlue device. Outbound Bluetooth connections can be established to either the Serial Port Profile (SPP) or the Dial-Up Networking Profile (DUN).

Syntax

con *address* [*profile*] [*timeout*]<CR>

Parameters

profile The Bluetooth profile to connect with. This parameter must be either "spp" (default) or "dun".

address The Bluetooth address of the remote device. The Bluetooth device address is the 48-bit IEEE address which is unique for each Bluetooth unit. The format of a Bluetooth device address is a series of six hexadecimal byte values separated by colons, i.e., 00:0C:84:00:05:29.

timeout An optional parameter used to abort the connection request after the specified number of seconds. The default timeout value is 15 seconds and the maximum value is 120 seconds.

Example

```
>con 00:0C:84:00:05:29<CR>
ACK<CR>
>
```

Example 2

```
>con 00:0C:84:00:05:29 dun 10<CR>
ACK<CR>
>
```

Delete Trusted Device

The *delete trusted device* command removes the remote device from the trusted list and prevents it from being able to connect with the local EmbeddedBlue device when security is turned on. A delete can be performed for either a single device by passing its device address or for all trusted devices by specifying the keyword all.

Syntax

del trusted all | *address*<CR>

Parameters

all This parameter is used to remove all devices from trusted status.

address The Bluetooth address of the device that should be removed from trusted status. The format of a Bluetooth device address is a series of six hexadecimal byte values separated by colons, i.e., 00:0C:84:00:05:29.

Example

```
>del trusted 00:0C:84:00:05:29<CR>
ACK<CR>
>
```

Disconnect

The *disconnect* command closes the connection with the remote Bluetooth device.

Syntax

dis<CR>

Example

```
>dis<CR>
ACK<CR>
>
```

Get Address

The *get address* command returns a globally unique address for either the local EmbeddedBlue device or the remote Bluetooth device.

Syntax

get address [local | remote]<CR>

Parameters

local	Return the local EmbeddedBlue device address (default).
remote	Return the remote Bluetooth device address.

Returns

The unique address of the local EmbeddedBlue device used to identify the device when making connections.

Example

```
>get address local<CR>
ACK<CR>
00:0C:84:00:05:29<CR>
>
```

Get Connectable

The *get connectable* command returns the current connectable setting of the local EmbeddedBlue device.

Syntax

get connectable<CR>

Returns

on	The device will accept connections (default).
off	The device will NOT accept connections.

Example

```
>get connectable<CR>
ACK<CR>
on<CR>
>
```

Get Encryption

The *get encryption* command returns the current encryption setting of the local EmbeddedBlue device. This setting controls whether transmitted data is encrypted or sent in the clear.

Syntax

get encrypt<CR>

Returns

on	Transmitted data will be encrypted (default).
off	Transmitted data will NOT be encrypted.

Example

```
>get encrypt<CR>
ACK<CR>
on<CR>
>
```

Get Escape Character

The *get escape character* command returns the current character used in the soft break command to instruct the local EmbeddedBlue device to break the flow of data and begin accepting commands. The escape character is '+' (ASCII 0x2B) by default.

Syntax

get escchar<CR>

Example

```
>get escchar<CR>
ACK<CR>
+<CR>
>
```

Get Flow Control

The *get flow control* command returns the flow control setting of the local EmbeddedBlue device.

Syntax

get flow<CR>

Returns

none	The EmbeddedBlue device is configured for no flow control (default).
hardware	The EmbeddedBlue device is configured for hardware flow control and the RTS line is used for receive flow control and the CTS line is used for transmit flow control.

Example

```
>get flow<CR>
ACK<CR>
none<CR>
>
```

Get Link Timeout

The *get link timeout* command returns the amount of time, in seconds, it takes for the local EmbeddedBlue device to notice that the connection has been broken if the remote device disappears. The link timeout is set to five seconds by default.

Syntax

get linktimeout<CR>

Example

```
>get linktimeout<CR>
ACK<CR>
5<CR>
>
```

Get Name

The *get name* command returns the name of the local EmbeddedBlue device. This is the value that is transmitted when a remote device performs an inquiry and then requests the device name. If you look for Bluetooth devices from a PC or PDA, this is the value that will be displayed to the user. The name is set to the device model number by default.

Syntax

get name<CR>

Example

```
>get name<CR>
ACK<CR>
eb100<CR>
>
```

Get Security

The *get security* command returns the current security setting of the local EmbeddedBlue device. When security is turned off, the device will allow connections to be established with any Bluetooth device. If the remote Bluetooth device provides the proper passkey, it will be added to the trusted list. When security is turned on, only devices in the trusted list will be allowed to connect.

Syntax

get security<CR>

Returns

on	Only devices in the trusted list will be allowed to connect.
off	Connections with any device will be allowed (default).

Example

```
>get security<CR>
ACK<CR>
off<CR>
>
```

Get Transmit Power

The *get transmit power* command returns the transmit power setting of the EmbeddedBlue radio. The transmit power is set to 10 (maximum) by default.

Syntax

get txpower<CR>

Returns

The current transmit power setting of the device. This will be an integer from 1 to 10.

Example

```
>get txpower<CR>
ACK<CR>
10<CR>
>
```

Get Visible

The *get visible* command returns the current visibility setting of the local EmbeddedBlue device. This setting controls whether the device can be seen by other Bluetooth devices.

Syntax

get visible<CR>

Returns

on	The device is visible to other devices (default).
off	The device is NOT visible to other devices.

Example

```
>get visible<CR>
ACK<CR>
on<CR>
>
```

Help

The *help* command will return a full listing of the available commands or a brief description of any individual command.

Syntax

hlp [*command*]<CR>

Parameters

command	The EmbeddedBlue command name for which to return help.

Examples

```
>hlp<CR>
ACK<CR>
… Help Information …
<CR>
>
```

```
>hlp connect<CR>
ACK<CR>
… Help Information on the Connect Command …
<CR>
>
```

List Trusted Devices

The *list trusted devices* command returns a list of all the devices that are allowed to connect to the local Bluetooth device when security is turned on. The maximum number of devices that can be trusted at any given time is twenty five; therefore, this command will return a list of between zero and twenty five addresses. When security is turned off, new devices can be added to this list by presenting the proper passkey when establishing a new connection.

Syntax

lst trusted<CR>

Returns

List of the trusted device addresses.

Example

```
>lst trusted<CR>
ACK<CR>
00:0C:84:00:05:29<CR>
00:80:C8:35:2C:B8<CR>
>
```

List Visible Devices

The *list visible devices* command returns a listing of all the Bluetooth devices that are currently in range and visible. The command may be canceled before the timeout is reached by sending an additional carriage-return to the device. Once the *list visible devices* command has been successfully canceled, a prompt (>) is issued by the EmbeddedBlue device.

Syntax

lst visible [name] [*timeout*]<CR>

Parameters

name	An optional parameter used to indicate the *list visible devices* command shall include the name of the remote Bluetooth devices in the response.
timeout	An optional parameter used to abort the *list visible devices* command after the specified number of seconds. This value must be a number from 1 to 120 seconds. The default value is 15 seconds.

Returns

The addresses of the Bluetooth devices that are in range and visible.

Example

```
>lst visible<CR>
ACK<CR>
00:0C:84:00:05:29<CR>
00:80:C8:35:2C:B8<CR>
```

```
>
```

Example 2

```
>lst visible name 15<CR>
ACK<CR>
00:0C:84:00:05:29 MyCellPhone<CR>
00:80:C8:35:2C:B8 JoesLaptop<CR>
>
```

Reset

The *reset* command will cause the local EmbeddedBlue device to reboot. A prompt (>) is issued by the EmbeddedBlue device once it is up and running again. An optional parameter, *factory*, will restore the factory default settings and then cause a reboot of the local EmbeddedBlue device.

Syntax

rst [factory]<CR >

Parameters

factory	Restores all device settings to factory defaults. This includes the baud rate parameter which may cause serial communications to be lost after the command is issued. To reestablish communications with the device after resetting to factory defaults, simply adjust the baud rate on the microprocessor serial port to match the device default rate of 9600bps.

Example

```
>rst factory<CR>
ACK<CR>
>
```

Return from Soft Break

The *return from soft break* command instructs the device to stop accepting commands and return to passing data wirelessly when there is an active connection.

Syntax

ret<CR>

Example

```
>ret<CR>
ACK<CR>
>
```

Set Baud Rate

The *set baud rate* command sets the baud rate for communications with the local EmbeddedBlue device.

Syntax

set baud *rate*<CR>

Parameters

rate The baud rate value. Valid baud rates are 1200, 2400, 4800, 9600 (default), 19200, 38400, 57600, 115200, 230400, and 460800. Once the baud rate has been set, applications, such as HyperTerminal, must also be configured to the same baud rate to continue communications.

Example

```
>set baud 19200<CR>
ACK<CR>
>
```

Set Connectable

The *set connectable* command provides control over whether the local EmbeddedBlue device will accept connections from other Bluetooth devices. In Bluetooth terminology, this command controls the setting for page scan.

Syntax

set connectable on | off<CR>

Parameters

on Configures the device so that other Bluetooth devices may establish a connection (default).

off Configures the device so that other Bluetooth devices may not establish a connection.

Example

```
>set connectable off<CR>
ACK<CR>
>
```

Set Encryption

The *set encryption* command provides control over whether transmitted data is encrypted or sent in the clear. EmbeddedBlue devices use 56-bit encryption to encrypt transmitted data when this setting is turned on.

Syntax

set encrypt on | off<CR>

Parameters

on Transmitted data will be encrypted (default).

off Transmitted data will NOT be encrypted.

Example

```
>set encrypt on<CR>
ACK<CR>
>
```

Set Escape Character

The *set escape character* command provides control over the character used in the soft break command to instruct the local EmbeddedBlue device to break the flow of data and begin accepting commands. The factory default escape character is the plus sign '+' (ASCII 0x2B).

Syntax

set escchar *character*<CR>

Parameters

character The character the device should recognize as the escape character in the soft break command.

Example

```
>set escchar &<CR>
ACK<CR>
>
```

Set Flow Control

The *set flow control* command provides control over the flow control setting of the local EmbeddedBlue device.

Syntax

set flow none | hardware<CR>

Parameters

none Configures the device for no flow (default).

hardware Configures the device for hardware flow control. The RTS line is used for receive flow control and the CTS line is used for transmit flow control.

Example

```
>set flow none<CR>
ACK<CR>
>
```

Set Link Timeout

The *set link timeout* command sets the amount of time it takes for the local EmbeddedBlue device to notice if the remote device disappears. In Bluetooth terminology, this command controls the setting for link supervisor timeout.

Syntax

set linktimeout *timeout*<CR>

Parameters

timeout The time, in seconds, it takes for the device to notice that a connection has been broken. The timeout may be set to any value between 1 and 40 and the default value is 5.

Example

```
>set linktimeout 10<CR>
ACK<CR>
>
```

Set Name

The *set name* command sets the name of the local EmbeddedBue device. This is the value that is transmitted when a remote device performs an Inquiry and then requests the device name. If you look for local Bluetooth devices from a PC or PDA, this is the value that will be displayed to the user.

Syntax

set name *value*<CR>

Parameters

value	A new device name. This value can be up to 32 characters in length and may contain any valid ASCII characters except for spaces (ASCII 0x20).

Example

```
>set name eb100<CR>
ACK<CR>
>
```

Set Passkey

The *set passkey* command sets the passkey that is used when establishing a connection with a new device. The passkey is set to 0000 by default, but this value can be changed to enhance security. It is recommended that you use a passkey that is 8 to 16 digits long.

Syntax

set passkey *value*<CR>

Parameters

value	A new passkey value that is between 1 and 16 digits long.

Example

```
>set passkey MyNewKey<CR>
ACK<CR>
>
```

Set Security

The *set security* command sets the current security setting of the local Bluetooth device. When security is turned off, the device will allow connections to be established with any Bluetooth device. If the remote Bluetooth device provides the proper passkey, it will be added to the trusted list. When security is turned on, only devices already on the trusted list will be allowed to connect.

Syntax

set security off | on<CR>

Parameters

on	Only connections with trusted devices will be allowed.
off	Connections with any device will be allowed (default).

Example

```
>set security on<CR>
ACK<CR>
>
```

Set Transmit Power

The *set transmit power* command sets the transmit power setting of the EmbeddedBlue radio. This command provides control over how much power the local EmbeddedBlue device will draw for transmitting radio frequencies. This power setting has a correlation to Bluetooth range: a lower transmit power setting will result in a shorter achievable Bluetooth range and it will also draw less power from the power source.

Syntax

set txpower *power*<CR>

Parameters

power	The power setting which is an integer between 1-10. Low values equate to a lower power setting. High values equate to a higher power setting. The default value is 10 (maximum).

Example

```
>set txpower 10<CR>
ACK<CR>
10<CR>
>
```

Set Visible Mode

The *set visible mode* command provides control over whether the local EmbeddedBlue device can be seen by other Bluetooth devices. In Bluetooth terminology, this command controls the setting for inquiry scan.

Syntax

set visible on | off<CR>

Parameters

on	Configures the device so that other Bluetooth devices can detect its presence (default).
off	Configures the device so that other Bluetooth devices can NOT detect its presence.

Example

```
>set visible on<CR>
ACK<CR>
>
```

Soft Break

The *soft break* command instructs the local EmbeddedBlue device to break the flow of data and begin accepting commands.

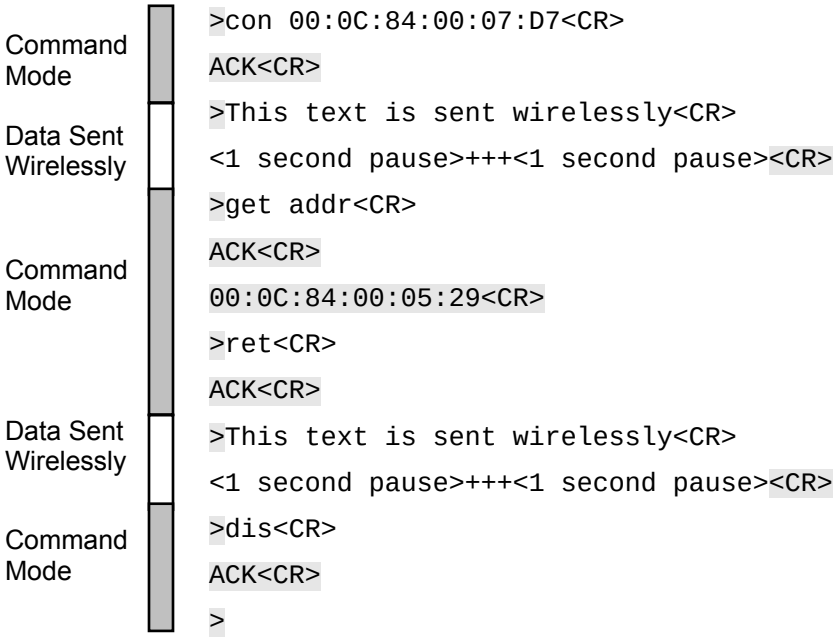
Syntax

<1 second pause>*esc sequence*<1 second pause>

Parameters

esc sequence	Three consecutive instances of the escape character. The factory default escape character is the plus sign (+). A different escape character can be set by using the Set Escape Character command.

Example



Version

The *version* command returns the current firmware version of the local EmbeddedBlue device.

Syntax

ver [*all*] <CR>

Parameters

all	An optional parameter used to return the firmware version, model number, and copyright information.

Example

```
>ver all<CR>
ACK<CR>
ebSerial 2.1.097<CR>
eb100-SER (00:0C:84:00:05:29)<CR>
(C) Copyright 2004-2007 A7 Engineering, Inc.<CR>
>
```

This page intentionally left blank.

Security

Bluetooth security is defined by three main elements: availability, access, and confidentiality. It is important to distinguish between these elements because Bluetooth security is also highly configurable so that it can meet the needs of devices in many different scenarios. An understanding of the basics will provide the knowledge that you need to choose a security strategy for your device.

The first important element of Bluetooth security is availability. If a device cannot be seen or connected with, it is obviously quite secure. Bluetooth defines both of these features as part of the security model and they are exposed by the ebSerial firmware through the *set visible* and *set connectable* commands. This is a very coarse level of control, but it is also quite effective and can be used in combination with other security features.

The second and most complex element of Bluetooth security is access control. This type of security is only relevant when the device is *connectable* and is designed to provide protection in this case. The general idea is that remote devices must become trusted before they will be allowed to connect and communicate with an EmbeddedBlue device. In order to become trusted, a remote device must present a passkey that matches the stored local passkey. This only needs to be done once, as both devices will remember their trusted status and allow future connections with that specific device without exchanging passkeys again.

The ebSerial firmware uses the *set security* command to configure access control. When security is turned *off*, connection attempts will be allowed from all remote devices. Forming a trusted relationship is carried out automatically in this mode the first time that a device attempts to establish a connection with the proper passkey. When security is turned on, only connections from trusted devices will be allowed and no new devices may become trusted.

The last element of Bluetooth security is confidentiality. Once a link with a trusted device has been established, it may be important to know that the data being transmitted cannot be intercepted by a third party. All transmitted data can be encrypted by configuring the *encrypt* setting to *on*. This only has an effect when connecting with other trusted devices.

This page intentionally left blank.

Error Codes

While using the ebSerial firmware you may encounter an error. Below is a listing of all ebSerial error codes with a description of what causes each error to occur.

Error Code	Description
1	General connection failure.
2	Connection attempt failed. This error occurs when attempting to connect with an invalid Bluetooth address or a device that is not available.
3	Command not valid while active. This error occurs when there is an active connection and a command is issued that is not valid while connected with a remote device.
4	Command only valid while active. This error occurs when there is not an active connection and a command is issued that is only valid while connected with a remote device.
5	An unexpected request occurred. This error occurs when the remote device makes an invalid request. This is typically seen with older Bluetooth devices that may have errors in their firmware.
6	Connection attempt failed due to a timeout.
7	Connection attempt was refused by the remote device. This error typically occurs when the security settings of the remote and local device are incompatible. It can also occur when establishing a connection if the local and remote passkeys do not match.
8	Connection attempt failed because the remote device does not support the Serial Port Profile.
9	An unexpected error occurred when deleting trusted devices.
10	Unable to add a new trusted device. This error will occur if you attempt to have more than twenty five simultaneously trusted devices.
11	Trusted device not found. This error occurs when the trusted device address is not recognized.
12	Command not valid during startup. This error occurs when a command has been issued before the EmbeddedBlue device is fully powered up and initialized.

Table 1: ebSerial Error Codes

This page intentionally left blank.

Contact Information

A7 Engineering has created the EmbeddedBlue product line of easy to use wireless solutions for embedded systems. In addition, A7 provides several levels of support for OEM product integration, certification, and even custom solutions.

Website: www.a7eng.com

Support Email: support@a7eng.com

Online Forum: http://www.a7eng.com/support/forum/forum.htm

Sales Email: sales@a7eng.com

A7 Engineering, Inc.

12127 Kirkham Road
Suite 101
Poway, CA 92064
858.391.1960 main
619.956.0082 fax

This page intentionally left blank.

Revision History

Date	Description
April 10, 2006	Initial publication of this document for ebSerial v2.0
March 26, 2007	Updated the documentation for ebSerial v2.1. Please refer to the firmware release notes for a comprehensive list of changes.