EmbeddedBlue™ 505

User Manual

Part Number 0000112 – Revision A

Last revised on June 28, 2005 – Printed in the United States of America

A7 Engineering, Inc.
12860 C Danielson Court
Poway, CA 92064

Life Support Policy and Use in Safety-Critical Applications:

A7's products are not authorized for use in life-support or safety-critical applications. Use in such applications is done at the sole discretion of the customer and A7 will not warrant or authorize the use of its devices in such applications.

Table of Contents

Table of Figures

Table of Tables

Introduction

Congratulations on your purchase of the EmbeddedBlue 505 (eb505) serial Bluetooth module. The eb505 is designed for direct connection to microcontrollers with a 5V serial interface enabling wireless communications with other Bluetooth devices including cellular phones, handheld computers, PCs, and other serial port adapters. Hobbyists, developers, and OEMs can take advantage of advanced wireless connectivity with this easy to use module.

The eb505 module provides a point to point connection much like a standard serial cable. Connections are made dynamically and can be established between two eb505 modules or an eb505 module and a standard Bluetooth v1.1, v1.2, or v2.0 device. Devices can be dynamically discovered and connected in an ad-hoc manner.

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 eb505.`
- `Inside a gray box is used to show data that is sent from the eb505.`

`Text in this font`

- `Is used to show source code`

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.
- Required parameter options are separated by a vertical bar |.
- Optional parameters are enclosed in brackets [ ].

Getting More Information

The Bluetooth website, www.bluetooth.com, contains the Bluetooth specification, profiles, and other documents relevant to Bluetooth.

General information regarding the eb505 module, 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.

A7 Engineering also provides professional design services on a contract basis to anyone requiring assistance with their design and/or development of Bluetooth products. For further information visit the A7 Engineering website www.a7eng.com/services/services.htm.

Bluetooth Overview

What is Bluetooth?

To put it simply, Bluetooth is a technology standard for electronic devices to communicate with each other using short-range radio. It is often referred to as a “cable replacement” technology, because it is commonly used to connect things, such as cameras, headsets, and mobile phones that have traditionally been connected by wires. Bluetooth is much more than simply a way to cut the cord between today’s existing electronic devices. It is an enabling technology that will take these devices to new levels of productivity and functionality and enable a whole new class of devices designed with communications and connectivity in mind.

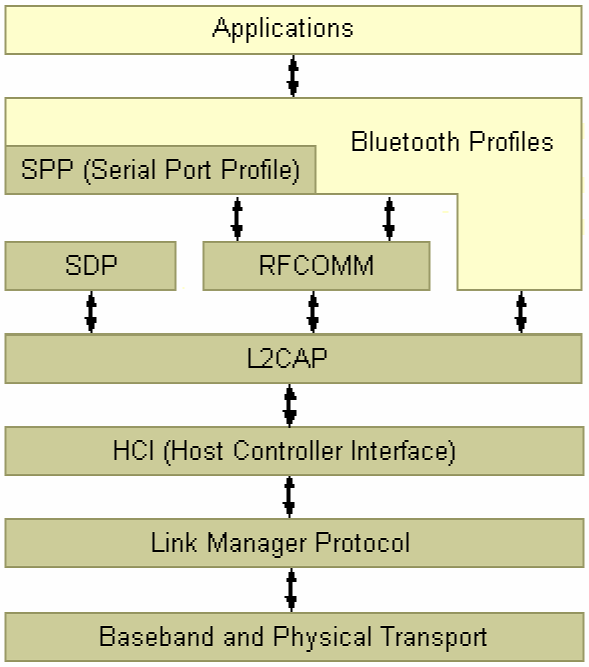
The Bluetooth Special Interest Group (SIG) defines Bluetooth a bit more broadly as the "worldwide specification for small-form-factor, low-cost radio solutions that provide links between mobile computers, mobile phones, other portable devices, and connectivity to the Internet." In defining Bluetooth, the SIG has taken a very different approach than the IEEE 802.11 Committees did. Rather than build Bluetooth as an adjunct to TCP/IP, it was defined as a standalone protocol stack that includes all layers required by an application. This means that it encompasses not only wireless communications but also service advertisement, addressing, routing, and a number of application-level interfaces referred to as profiles.

Bluetooth is based on a frequency hopping spread spectrum (FHSS) modulation technique. The term spread spectrum describes a number of methods for spreading a radio signal over multiple frequencies, either simultaneously (direct sequence) or in series (frequency hopping.) Wi-Fi devices are based on direct sequence spread spectrum transmission which uses multiple channels simultaneously. While this technique increases the speed of transmission (for example in Wi-Fi from 1.5MHz to 11MHz), it is more susceptible to interference from other radio sources as well as being a greater source of interference to the surrounding area.

In contrast, Bluetooth utilizes the frequency hopping method of spread spectrum which uses multiple radio channels to reduce interference and increase security. The signal is rapidly switched from channel to channel many times per second in a pseudo-random pattern that is known by both the sender and receiver(s). This provides robust recovery of packet errors caused by interference from another radio source at a particular frequency. Also, data is generally more secure because it is not possible to receive more than a fraction of the data unless the hopping pattern is known. Bluetooth utilizes frequency hopping in the 2.4GHz radio band and hops at a relatively fast pace with a raw data rate of about 1 Mbps. This translates to about 700 kbps of actual useful data transfer. The eb505 module supports a maximum sustained bidirectional data rate of 230.4kbps.

What is a Profile?

Bluetooth devices can support interoperability with one or more types of devices. In order for two Bluetooth devices to communicate with each other, they must share at least one common profile. If I want a Pocket PC to communicate with my EmbeddedBlue radio I need to make sure that they both support the same profile. EmbeddedBlue devices support the Serial Port Profile (SPP) which is one of the earliest and most widely supported profiles.


Applications
Bluetooth Profiles
SPP (Serial Port Profile)
SDP
RFCOMM
L2CAP
HCI (Host Controller Interface)
Link Manager Protocol
Baseband and Physical Transport

The main elements of the Bluetooth stack are shown in the figure to the right. As with a typical diagram of the TCP/IP stack, there are a number of details that are hidden by the apparent simplicity of the stack. Specifically, there are a number of profiles that sit roughly on top of the L2CAP layer that provide much of the power (and also the complexity) of the Bluetooth protocols.

These profiles are the primary entry into the stack for an application. Essentially, they define the set of services that are available to that application. Currently there are more than 25 different profiles defined or in the process of being defined by the Bluetooth SIG. With so much variety, acquiring an in-depth understanding of Bluetooth is not a trivial task. However, the abstraction by a single profile can provide an application the use of the profile without such detailed knowledge.

There are a number of profiles that are exposed in very familiar forms. The eb505 module, for instance, implements the SPP profile which enables it to appear like a traditional serial port. This virtually eliminates the need for the user to have specific Bluetooth knowledge and allows the radios to be integrated into applications very quickly.

Bluetooth and Wi-Fi

Bluetooth and Wi-Fi are often compared to each other because they are both capable of providing networking on the 2.4GHz consumer frequency band. Many of the differences between these two technologies can be traced to the fact that networking was not the primary design goal for Bluetooth as it was for Wi-Fi. With a greater transmission range (about 100 meters indoors) and larger bandwidth (about 11Mbps), Wi-Fi is typically the better choice for wireless LANs and Internet connectivity.

Bluetooth on the other hand was designed for driverless, cordless connectivity between devices. Because Bluetooth transmitters are smaller in size, have lower power demands, a more limited range (10 - 100 meters) and narrow bandwidth (1Mbps), they are better suited for use in embedded and mobile devices that exchange smaller amounts of information while conserving power and space.

While their functionality does not compete directly, 802.11b and Bluetooth do compete for the airwaves. Since they both operate on the 2.4GHz band of the ISM radio spectrum, these two wireless technologies may interfere with each other. Bluetooth devices minimize interference by employing a frequency-hopping spread spectrum scheme that changes the frequency used about 1600 times per second. Unfortunately, since Wi-Fi uses a direct sequence spread spectrum method, this also means that Bluetooth transmissions will collide with those of any nearby 802.11b devices and slow Wi-Fi data transmission rates. The Bluetooth SIG and its member companies have put a lot of effort into coexistence solutions for these two standards and are very committed to ensuring that these devices work well together.

While 802.11b was designed solely for data communications, Bluetooth takes things quite a bit further. A key component of the Bluetooth standard is its notification and service discovery mechanism. This allows Bluetooth devices to identify themselves and describe their capabilities to other Bluetooth devices in the area. For instance, the Dial-Up Networking profile defines how discoverability can be used to locate and connect to other devices such as a cellular phone that supports the same profile. The profile then describes how to dial the phone, connect to either analog or data services, and control the connection seamlessly. This combination of dynamic discovery of services and built in definitions of the services goes well beyond anything offered by the 802.11b protocol.

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 EmbeddedBlue device 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 module 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 the EmbeddedBlue module. 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 EmbeddedBlue module uses the *set security* command to configure access control. There are three possible settings for security, *off*, *open*, and *closed*. When security is turned *off*, connection attempts will be allowed from all remote devices. When security is set to *open,* connections are only allowed from trusted devices, but new devices can become trusted by presenting the correct passkey. Forming a trusted relationship is carried out automatically in this mode the first time that a remote device connects with the EmbeddedBlue module. When security is set to *closed*, only connections from trusted devices will be allowed and no new devices may become trusted. Closed security is the most restrictive setting and therefore the most secure.

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 security is set to either *open* or *closed*. The EmbeddedBlue module supports 56-bit encryption by default, but 128-bit encryption is available. Due to export restrictions to certain countries, firmware supporting 128-bit encryption is only available with proper approval from A7 Engineering.

The Basics

Most of the complexity of working with Bluetooth has been encapsulated in the EmbeddedBlue module in order to make it easier to use. The specific 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 set up, it is a simple matter to communicate between the endpoints of that connection using familiar and well-supported programming constructs.

Command Mode

The eb505 supports two main operating modes: command mode and data mode. Upon power up, the eb505 enters command mode and is ready to accept serial commands. The factory default communication parameters are 9600 Baud, 8 Data Bits, 1 Stop Bit, No Parity, and No Flow Control. The eb505 supports commands to modify the baud rate and flow control settings.

In this mode 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 eb505 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 13) is used to mark the end of the command.

Data Mode

Once the eb505 radio is connected to another Bluetooth device, the eb505 automatically switches into data mode. All data transmitted while in this mode will be sent to the remote device and, therefore, NO further commands can be sent until the eb505 radio is disconnected or switched back to command mode by use of the mode control I/O line or the Switch to Command Mode sequence.

The connection status line of the eb505 module can be monitored to determine if there is an active connection. Additionally, whenever a connection is present, the Connection Status LED on the eb505 module will be on.

Resetting the eb505 to the Factory Default Settings

There are two different mechanisms to reset the eb505 module to the factory default settings. Either by shorting the STATUS and MODE pins (pin 8 and pin 9) and then applying power to the eb505 module, or by issuing the reset command to the eb505; see the command set reference at the back of this manual for the syntax of the reset command.

Switching between Data Mode and Command Mode

When a Connection command is issued, the eb505 attempts to establish a connection to the device with the address specified in the command. Once a connection is established, the eb505 switches into data mode. At this point all data sent to the eb505 is transmitted to the remote Bluetooth device over the wireless link. It is possible to switch from data mode to command mode, issue commands, and then return to data mode, while maintaining a connection. The eb505 allows you to switch between data mode and command mode by issuing the Switch to Command Mode and Return to Data Mode commands or by driving the MODE control I/O line (P6) of the eb505 module.

Hardware Connections

The eb505 module is designed to interface with a 5V signal environment. It supports a power supply of 5 – 12V and can be connected directly to 0.1” breadboards and 10 x 2 sockets. All of the required connections are contained on header CN1. In the diagram below, Pin 1 of connector CN1 is marked with a white dot (Figure 1). A full device pin out is available in the Technical Specifications section of this manual.

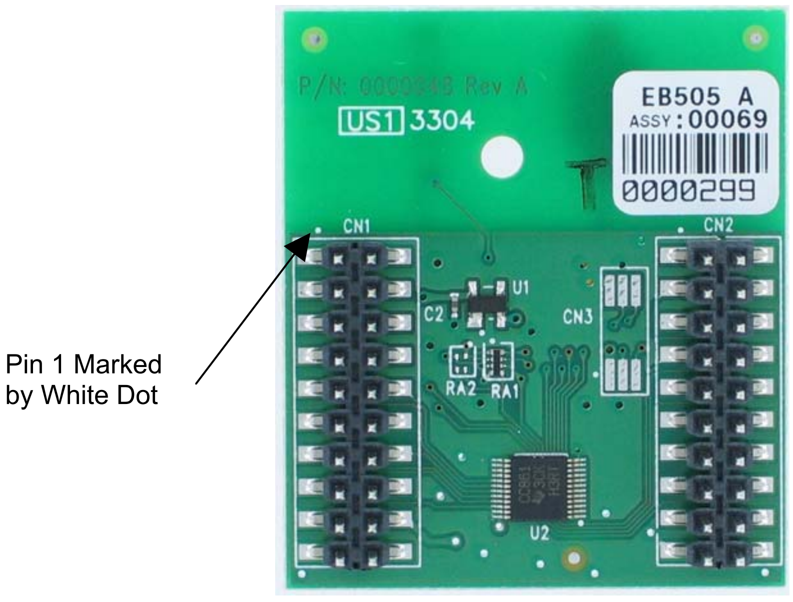


Figure 1: eb505 Module

This page intentionally left blank.

Command Set

The EmbeddedBlue command set 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 program, written in a programming language such as C++ or Visual Basic, running on a PC, using an RS232 adapter. From a microcontroller application, these commands can be issued directly using the provided serial port functions.

Command Basics

Commands may only be sent to the module when it is in Command Mode. White spaces are used to separate parameters of the command and a carriage-return is used to mark the end of the command. Upon receipt of a command the eb505 begins to parse the parameters. If the syntax of the command is correct the eb505 returns an ACK string, not the ACK character (0x06); otherwise, a NAK string is returned. Following the ACK or NAK string is a carriage-return (0x0D) character. If an error occurs while processing the command an error string is returned followed by a carriage-return followed by the prompt (>) character. If the command executed successfully the module will issue the prompt (>) character. Please see the Error Codes section for a description of the error codes.

The following example shows the basic structure of a command. A prompt (>) is issued by the EmbeddedBlue module. A command followed by a carriage-return is sent to the module. The module responds with either an ACK or NAK string followed by a carriage-return. If an error occurs, the module 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 module.

```
>command<CR>

ACK | NAK<CR>

Err number<CR>

>
```

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 the connection request.

Syntax

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

Parameters

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 maximum value is 120 seconds.

Example

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

ACK<CR>

>
```

Delete Trusted Device

The *delete trusted device* command removes the remote device from trusted status and prevents it from being able to connect with the EmbeddedBlue device when security is set to closed. 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 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.

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 the address of the local EmbeddedBlue device.

Syntax

get address<CR>

Returns

The unique address of the local EmbeddedBlue device used to identify the module when making connections. In Bluetooth terminology this is the Bluetooth Device Address.

Example

```
>get address<CR>

ACK<CR>

00:0C:84:00:05:29<CR>

>
```

Get Connectable Mode

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

Syntax

get connectable<CR>

Returns

The current connectable mode setting of the local EmbeddedBlue device. In Bluetooth terminology, the returned value reflects the current setting for page scan.

on	The device will accept connections.
off	The device will NOT accept connections.

Example

```
>get connectable<CR>

ACK<CR>

on<CR>

>
```

Get Encrypt Mode

The *get encrypt mode* command returns the current encryption setting of the module. This setting controls whether the module encrypts transmitted data when security is set to either open or closed. By default EmbeddedBlue modules use 56-bit encryption, but 128-bit encryption is available upon request. Contact A7 Engineering for more information about getting 128-bit encryption.

Syntax

get encrypt<CR>

Returns

The current encrypt mode setting of the module.

on	If security is set to open or closed, transmitted data will be encrypted.
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 Switch to Command Mode command to instruct the EmbeddedBlue device to leave Data Mode and enter Command Mode.

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 and both the RTS and CTS lines are configured as high Z inputs. An application is free to use these lines as normal I/O.
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. This timeout also has an effect on how robust the communications link is to interference. If this value is set very low, the link may be lost if interference picks up for several seconds, such as when a heavy burst of 802.11 traffic is encountered.

Syntax

get linktimeout<CR>

Example

```
>get linktimeout<CR>

ACK<CR>

5<CR>

>
```

Get Name

The *get name* command returns the name of the local 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

get name<CR>

Example

```
>get name<CR>
ACK<CR>
eb505<CR>
>
```

Get Security Mode

The *get security mode* command returns the modules current security mode setting. When security is turned off, the module will allow connections to be established by any Bluetooth device. When security is set to open, the remote Bluetooth device is required to provide a valid passkey before a connection can be established. When security is set to closed, only existing trusted devices are allowed to establish connections.

For maximum security it is recommended that the module be operated in closed mode whenever possible.

Note: The Security mode is not applicable if connectable mode is set to off.

Syntax

get security<CR>

Returns

The current security mode setting of the module.

off	The module will allow any Bluetooth device to establish a connection.
open	The module will allow any Bluetooth device that provides the correct passkey to establish a connection.
closed	The module will only allow trusted devices to establish a connection.

Example

```
>get security<CR>

ACK<CR>

open<CR>

>
```

Get Visible Mode

The *get visible mode* command returns the modules current visibility setting. This setting controls whether the module can be seen by other Bluetooth devices.

Syntax

get visible<CR>

Returns

The current visible mode setting of the module. In Bluetooth terminology, the returned value reflects the current setting for inquiry scan.

on	The module is visible to other devices.
off	The module is NOT visible to other devices.

Example

```
>get visible<CR>

ACK<CR>

on<CR>

>
```

Help

The *help* command returns a listing of the EmbeddedBlue commands and a brief description of each command.

Syntax

hlp [*command*]<CR>

Parameters

command	The EmbeddedBlue command name (con, del, dis, get, lst, rst, set, and ver) for which to return help.

Examples

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

>hlp con<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 when security is set to closed. The maximum number of devices that can be trusted at any given time is twenty five, so this command will return a list of between zero and twenty five addresses. When security is set to open, new devices can be added to this list by presenting the proper passkey while establishing a new connection.

Syntax

lst trusted<CR>

Returns

List of the trusted device addresses. These devices are the only ones that are allowed to connect with this module when security is set to closed.

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 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 module.

Syntax

lst visible [*timeout*]<CR>

Parameters

timeout	An optional parameter used to abort the list request after the specified number of seconds. The default value is 30. The maximum value is 120 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>

>
```

Reset Factory Defaults

The *reset factory defaults* command restores all module 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 module, simply adjust the baud rate on the microprocessor serial port to match the module default rate of 9600bps.

Syntax

rst factory<CR >

Parameters

factory	Resets all settings to factory defaults.

Example

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

Return to Data Mode

The *return to data mode* command instructs the module to enter Data Mode 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 module.

Syntax

set baud *rate* [*]<CR>

Parameters

rate	The baud rate value. Valid baud rates are 9600 (default), 19200, 38400, 57600, 115200, and 230400. Once the baud rate has been set, applications, such as HyperTerminal, must also be configured to the same baud rate to continue communicating with the eb505.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set baud 19200<CR>

ACK<CR>

>
```

Set Connectable Mode

The *set connectable mode* command provides control over whether the local EmbeddedBlue module 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 module so that other Bluetooth devices may establish a connection.
off	Configures the module so that other Bluetooth devices may not establish a connection.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set connectable off<CR>

ACK<CR>

>
```

Set Encrypt Mode

The *set encrypt mode* command provides control over whether transmitted data is encrypted or sent in the clear. This setting is only in effect when security is set to either open or closed. When security is turned off, the transmitted data is never encrypted. By default EmbeddedBlue modules use 56-bit encryption, but 128-bit encryption is available upon request. Contact A7 Engineering for more information about getting 128-bit encryption.

Syntax

set encrypt on **|** off [*]<CR>

Parameters

on	Configures the module so that transmitted data will be encrypted when security is set to either open or closed.
off	Configures the module so that transmitted data will NOT be encrypted.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

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

Set Escape Character

The *set escape character* command provides control over the character used in the Switch to Command Mode command to instruct the module to leave Data Mode and enter Command Mode. The factory default escape character is the plus sign (+).

Syntax

set escchar *character* [*]<CR>

Parameters

character	The character the module should recognize as the escape character used in the Switch to Command Mode command.
*	An optional parameter used to persist the new setting when the module is powered down.

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 module.

Syntax

set flow none **|** hardware [*] <CR>

Parameters

none	Configures the module for no flow control and both the RTS and CTS lines are configured as high Z inputs. This allows an application to use these lines a normal I/O.
hardware	Configures the module for hardware flow control. The RTS line is used for receive flow control and the CTS line is used for transmit flow control.
*	An optional parameter used to persist the new setting when the module is powered down.

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 module to notice that the connection has been broken, if the remote device disappears. This timeout also has an effect on how robust the communications link is to interference. If this value is set very low, the link may be lost if interference picks up for several seconds, such as when a heavy burst of 802.11 traffic is encountered. 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 module to notice that a connection has been broken. The default value is 5. The maximum value is 40 seconds.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

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

Set Name

The *set name* command sets the name of the local 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 character.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set name eb505<CR>

ACK<CR>

>
```

Set Passkey

The *set passkey* command sets the passkey that is used when establishing a connection with security set to open. The passkey is set to 0000 by default, but this value should 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.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set passkey MyNewKey<CR>

ACK<CR>

>
```

Set Security Mode

The *set security mode* command sets the module's current security mode setting. When security is turned off, the module will allow connections to be established by any Bluetooth device. When security is set to open, the remote Bluetooth device is required to provide a valid passkey before a connection can be established. When security is set to closed, only existing trusted devices are allowed to establish connections.

For maximum security it is recommended that the module be set to closed mode whenever possible.

Syntax

set security off | open | closed [*]<CR>

Parameters

off	Turns security off allowing any Bluetooth device to establish a connection.
open	Configures the module to require other devices to provide the correct passkey before establishing a connection.
closed	Configures the module to only allow trusted devices to establish a connection.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set security open<CR>
ACK<CR>
>
```

Set Visible Mode

The *set visible mode* command provides control over whether the module 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 module so that other Bluetooth devices can detect its presence.
off	Configures the module so that other Bluetooth devices can NOT detect its presence.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set visible on<CR>

ACK<CR>

>
```

Switch to Command Mode

The *switch to command mode* command instructs the EmbeddedBlue module to enter Command Mode.

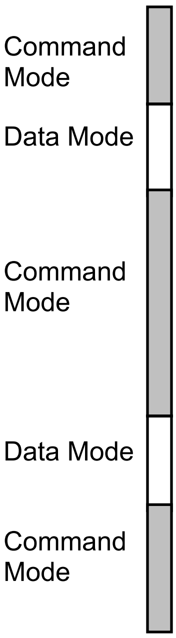
Syntax

<2 second pause>*esc sequence*<2 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



```
>con 00:0C:84:00:07:D7<CR>
ACK<CR>
>This text is sent in data mode<CR>
<2 second pause>+++<2 second pause><CR>
>get addr<CR>
ACK<CR>
00:0C:84:00:05:29<CR>
>ret<CR>
ACK<CR>
>This text is sent in data mode<CR>
<2 second pause>+++<2 second pause><CR>
>dis<CR>
ACK<CR>
>
```

Version

The *version* command returns the current firmware version of the EmbeddedBlue module.

Syntax

ver [all] <CR>

Parameters

all	An optional parameter used to return the build number, model number, serial number, and manufacturer.

Example

```
>ver all<CR>

ACK<CR>

Firmware Version: 2.0<CR>

Firmware Build: 247<CR>

Model Number: eb505<CR>

Serial Number: 1008<CR>

Manufacturer: A7 Engineering<CR>

>
```

Firmware Upgrade

From time-to-time A7 Engineering provides new versions of firmware that provide enhancements to the product. A7 Engineering also provides an EmbeddedBlue DFU utility which provides the mechanism to upgrade the firmware in the eb505 module. The latest version of the EmbeddedBlue DFU utility and firmware for the eb505 module may be obtained from the A7 Engineering web site at www.a7eng.com.

Upgrading the eb505 Firmware

This procedure will step you through the process of upgrading the firmware of the eb505 module. To upgrade the firmware of your eb505 you will need an RS232 adapter, a PC running Windows XP with Service Pack 2, the EmbeddedBlue DFU utility and the firmware upgrade file (a .DFU file obtained from the A7 Engineering web site).

Step 1: Install the EmbeddedBlue DFU Utility

In this step we will install the EmbeddedBlue DFU utility onto your PC. To execute this step you must have previously downloaded the EmbeddedBlue DFU utility from the A7 Engineering web site. If you have previously installed the EmbeddedBlue DFU utility you may proceed to the next step.

1. Navigate to the **folder** where you downloaded the EmbeddedBlue DFU utility and double-click on the **EmbeddedBlueDFU.exe**.
2. Step through the **install wizard** supplying the requested information on each of the dialogs of the wizard.

   It is recommended that you use the default settings of the install wizard.

Step 2: RS232 Level Translation Setup

In this step we will attach an eb505 module to the RS232 Adapter and apply power to the device.

1. Please follow the guidelines provided with the RS232 adapter to establish the connection.

Step 3: eb505 Module Setup

In this step we will configure the eb505 module to communicate with the EmbeddedBlue DFU Wizard. If your eb505 module is already configured to communicate at 9600 baud you may proceed to the next step.

1. Using a **terminal emulator**, such as HyperTerminal, establish a **connection** with the **eb505** attached to the RS232 Adapter.
2. Set the **baud rate** of the **eb505** module to **9600** baud by issuing the set baud rate command.

   To set the baud rate, type `set baud 9600` at the "`>`" prompt and press the return key.

   Example:

   ```
   >set baud 9600
   ACK
   >
   ```
3. Close the **terminal emulator**.

Step 4: Run the EmbeddedBlue DFU Wizard

In this step we will step through the EmbeddedBlue DFU Wizard and upgrade the firmware of the eb505 module.

1. Launch the **EmbeddedBlue DFU** Wizard.

   From the Start menu select All Programs then A7 Engineering then EmbeddedBlue DFU.
2. The first page of the wizard is the introduction page; click **Next** to continue.
3. On the **Connection Type** dialog, select **COM port (RS-232)**.
4. Click **Next.**

   The wizard will now search for available COM ports.
5. Select the **COM port** to which the **RS232 Adapter** is connected.
6. Click **Next.**
7. On the **Upgrade File** dialog, click **Browse** to navigate to the file containing the firmware upgrade.
8. On the **Select EmbeddedBlue Firmware File** dialog, select the **file containing the firmware upgrade** (a .DFU file).

9. Click **Select**.
10. On the **Upgrade File** dialog, click **Next**.

    This will display the Ready to Upgrade dialog.
11. Review the **information** shown on the **Ready to Upgrade** dialog.
12. Click **Next**.

    This will display the Upgrade in Progress dialog and the upgrade process will begin. When the upgrade is complete the Successful Upgrade dialog will appear.

> **You must not stop the upgrade process or remove power from the eb505 module until the upgrade is complete. If the upgrade process is interrupted the eb505 module may become non-functional.**

13. Review the **information** shown on the **Successful Upgrade** dialog.

    The firmware of your eb505 module has been upgraded.

    If the **Upgrade Failed** dialog appears click **Details…** to get additional information about the failure. Ensure that the eb505 is correctly inserted into the RS232 Adapter, that the RS232 Adapter is connected to the PC using the provided straight through serial cable and that you have selected the appropriate COM port in the EmbeddedBlue DFU wizard.
14. Click **Finish**.

Step 5: Check the Firmware Version

While it is not necessary to check the firmware version to complete the firmware upgrade process, this step demonstrates how to check the version of firmware on your EmbeddedBlue module.

1. Using a **terminal emulator**, such as HyperTerminal, establish a **connection** with the **eb505** attached to the RS232 Adapter.
2. Check the **version** information by issuing the version command.

   To view the version information, type `ver all` at the “>” prompt and press the return key.

   Example:

```
>ver all
```

```
ACK
Firmware Version: 2.0
Firmware Build: 247
Model Number: eb505
Serial Number: 1008
Manufacturer: A7 Engineering
>
```

3. Close the **terminal emulator**.

Error Codes

While using the eb505 you may encounter an error. Below is a listing of all eb505 error codes with a description of what causes the 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 with security set to open if the remote and local 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 module is fully powered up and initialized.

Table 1: eb505 Error Codes

This page intentionally left blank.

Technical Specifications

Operating Parameters

The operating parameters of the eb505 are shown below in Table 2.

Transmit Power	4dBm (max) class 2 operation	
Open Field Range	eb505-AHC-IN (surface mount antenna) – 10 meters (32 feet) *(Actual range is dependent upon location and environment.)*	
Receiver Sensitivity	-85dBm	
Operating Temp.	-15° to 70°C	
Supply Power	5 to 12VDC	
Current Consumption	115.2kbps data transfer: 35mA 38.4kbps data transfer: 30mA 9.6kbps data transfer: 25mA	connected and idle: 8mA no connection: 3mA
Interfaces	5V TTL UART Baud rate 9.6k – 230.4k Flow control: RTS/CTS or none	
Connector	Two 10x2 20 pin 0.1” headers	
Antenna	Matched internal surface mount	
Bluetooth Support	Version 1.2 compliant with profiles L2CAP, RFCOMM, SDP, SPP	
Firmware	Upgradeable via PC application with RS232 adapter.	

Table 2: eb505 Operating Parameters

Dimensions

The dimensions of the eb505 are shown below in Table 3. Please reference Figure 2 to locate the referenced dimension on the eb505.

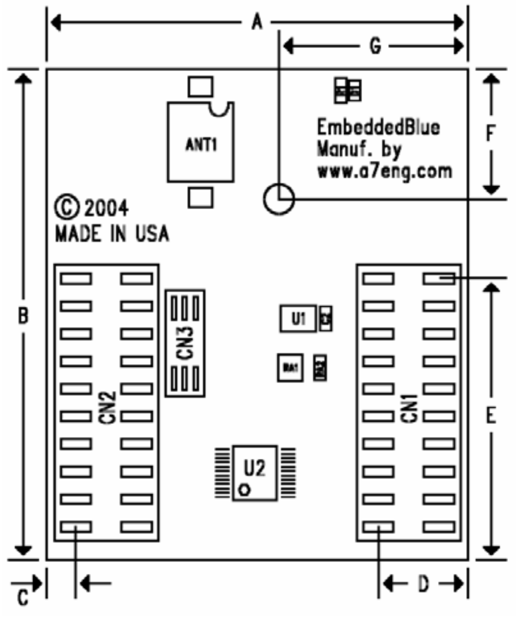


Figure 2: eb505 Dimensions

Dimension	inches	mm
A	1.54	39.00
B	1.77	44.90
C	0.17	4.23
D	0.27	6.78
E	1.02	25.90
F	0.45	11.40
G	0.67	17.00

Table 3: eb505 Dimensions

Pin out

The eb505 module features two 10x2 20 pin connectors with 0.1” spacing. Currently, ten of the pins are in use (eight when flow control is set to none). The other pins are reserved for future use.

Pin	Function	Type	Description	Usage
CN1 - 2	GND		Ground	Required
CN1 - 4	TX	TTL	Serial Transmit line from eb505; TTL	Required
CN1 - 6	RX	TTL	Serial Receive line to eb505	Required
CN1 - 8	Status	TTL	Bluetooth connection status (0 = not connected, 1 = connected)	Required
CN1 - 10	Mode	TTL	Command/data mode toggle (0 = command, 1 = data)	Required
CN1 - 12	RTS	TTL	Request-to-Send used for hardware flow control (0 = not signaled, 1 = signaled)	Optional
CN1 - 14	CTS	TTL	Clear-to-Send used for hardware flow control (0 = not signaled, 1 = signaled)	Optional
CN1 - 18	On/Off	TTL	Powers the eb505 up or down (0 = off, 1 = on) weak internal pull-up	Required
CN1 - 20	VCC		Power	Required
CN2 – 19	GND		Ground	Required

Table 4: eb505 Pin out Description

This page intentionally left blank.

Frequently Asked Questions

Question: How do I obtain eMbedded Visual C++ 4.0 to develop Pocket PC applications?

Answer: The eMbedded Visual C++ 4.0 development tool is available from Microsoft. In addition, you will need eMbedded Visual C++ 4.0 SP2 and the SDK for Windows Mobile™ 2003-based Pocket PCs. These tools can be downloaded free of charge from the Microsoft Windows Mobile web site: http://www.microsoft.com/windowsmobile.

Question: Why is my eb505 not displayed when I try to discover it from my PC or Pocket PC?

Answer: Verify that the eb505 module is properly powered. It is likely you will discover the eb505 on the first attempt; however, because Bluetooth discovery is not deterministic, discovery on the first attempt is not guaranteed. On the PC or Pocket PC, use the refresh option to search for devices again. Verify that the visible mode setting in the eb505 is set to on.

Question: I can discover my eb505, but why am I unable to establish a connection?

Answer: Verify that the connectable mode setting in the eb505 is set to on and that security is set either to off or open. In closed security mode only devices that have already established a trusted relationship will be allowed to connect.

Question: When I try to connect from an EmbeddedBlue device with 1.0 firmware to one with 2.0 firmware the connection attempt times out and then fails with Error 2. Why?

Answer: Version 1.0 firmware did not support passkey security and trusted relationships, which is enabled as the default in version 2.0 firmware. To connect from a version 1.0 device you will need to disable security on the version 2.0 device with the “set security off” command.

Question: I used the *set visible* command to make the eb505 module not visible to other devices, but when I perform a scan from my PC I still see the device. Why?

Answer: Most of the PC Bluetooth implementations cache device scan results to save time. If you located the eb505 module before making it invisible, the PC will remember the device even though it can no longer be seen. These results are typically only cached until the Bluetooth stack is reset, so if you reboot the PC or remove and reinsert the dongle you should no longer see the device.

Contact Information

A7 Engineering has created the EmbeddedBlue product line of easy to use wireless solutions for 8 and 16 bit 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.

12860 C Danielson Court
Poway, CA 92064
858.679.7708 main
858.391.5616 fax