A7 eb101 Serial Firmware Version 1.0.025
A7 eb101/eb301 Bluetooth Serial Devices

Quick Start Guide

Revised June 16, 2009

Notes

Introduction

A7's eb101 serial firmware provides the basis for the functionality of the eb101 module family for use in scenarios such as cable replacement. In turn, a number of A7's Bluetooth adapters, including the eb301 adapters, feature the eb101 module as the core of the adapter.

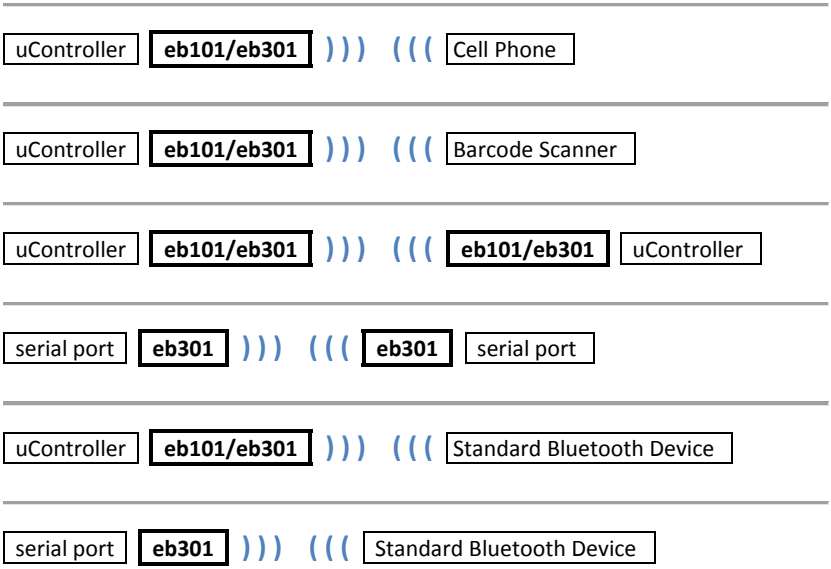
This quick reference guide provides an overview of the command interface and other quick reference information for the serial firmware and the hardware running that firmware.

A7's eb101 serial firmware is designed to abstract the details of Bluetooth and to make setup and connection to other eb101 serial firmware products as well as other standard Bluetooth devices a simple task.

The firmware supports two main operating modes: EasyConnect mode and command mode. EasyConnect mode is generally used in simple cable replacement scenarios; while command mode provides a rich set of functions that allow programmatic control.

The eb101 module is certified as a Bluetooth end product with the Bluetooth SIG. The Bluetooth qualified device ID is **B014985**. The module is also FCC part 15 pre-certified, with a FCC ID of **WND-1000160**.

The following illustrates some of the many device scenarios for the eb101 module and eb101 module based adapters such as the eb301 adapters.



UARTs & RS232: A Primer/Refresher

General Overview

UARTs and RS232 are used for asynchronous serial communications. The format of the serial link is specified using a start bit, data bits, a parity bit (or not), and a stop bit(s). The timing of the bits is defined by a baud rate. Being asynchronous, it is up to each side to internally clock the data from detection of the start bit within the inherent tolerance dictated by the baud rate. It is therefore also important that both sides of the link have the same configuration for format.

Details

The eb101 and eb301 use a default configuration of:

- 8 data bits (not configurable)
- No parity (configurable as odd or even also)
- 1 stop bit (which means a single bit in duration; note that this is not configurable)
- 9600 baud (configurable at standard baud rates from 1200 to 460800 baud)

RS232 is specified to use voltage levels that are not at the same levels as standard digital logic signals typically used for UARTs. In fact, they are specified as exceeding +3V for a logic '0' and falling below -3V for a logic '1'. Any voltage level between these two levels is specified as undefined. In order to convert normal digital logic levels to these RS232 levels, special driver chips are commonly used. Often these chips are referred to as RS232 line drivers or MAX-232 type devices. The standard digital logic is often referred to as a UART interface; and the interface after being converted to RS232 is, of course, referred to as a RS232 interface.

Often UARTs are implemented on specific pins in microcontrollers; however they are also often implemented via "bit banging". "Bit banging" simply means to toggle a port pin in firmware for transmitting and to read a port pin in firmware for receiving; the timing of which is handled by the firmware to achieve the desired baud rate.

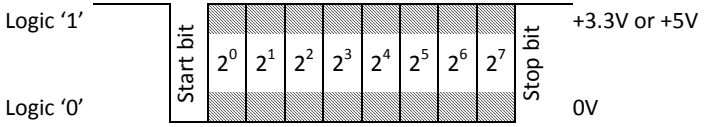
The eb101 and the 3V/5V version of the eb301 do not use RS232; but rather, are made for direct connection to UARTs, such as port pins on a microcontroller. In contrast the RS232 version of eb301 does use RS232 and therefore is designed with an onboard RS232 line driver/receiver for connection to RS232 level devices; such as that of a PC serial port or device that might connect to a PC serial port.

Illustrations

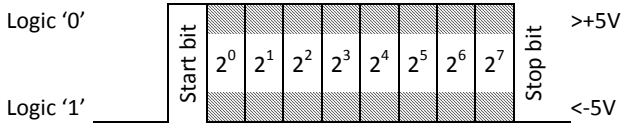
The following two examples illustrate the difference between the UART interface and the RS232 interface. Note:

- The bits are inverted; however the logic remains the same. (E.g. a start bit is represented with a logic '0'). This applies to all bits, including data bits.
- In both examples the parity bit does not exist, as the default for the eb101 and eb301 products is "no parity".
- The state of each of the 8 bits of data is determined by the actual data being sent.
- The data payload is transmitted, least significant bit first.

*Here is an example of the TX or RX line for an **eb101 or 3V/5V eb301**. (UART interface)*



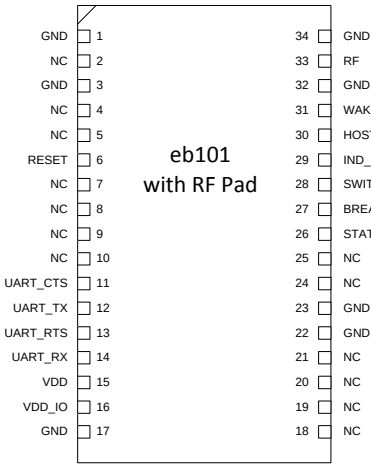
*Here is an example of the TX or RX line for a **RS232 eb301**. (RS232 interface)*

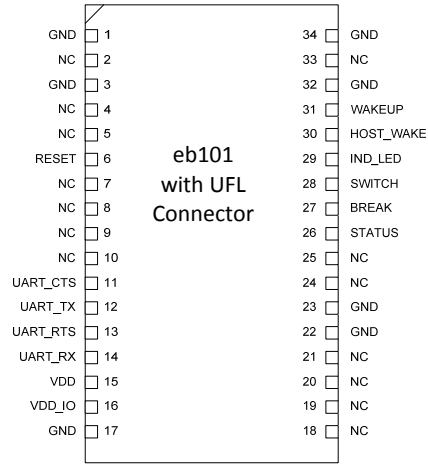


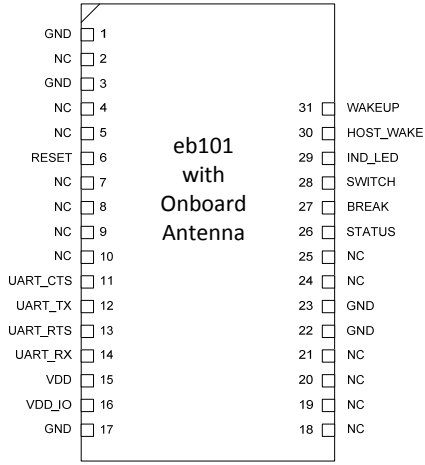
Note that in addition to the serial data TX and RX signals the interface also allows for optional hardware handshaking signals, CTS and RTS.

A7 eb101 Bluetooth Serial Modules

For detailed information about the eb101 hardware please refer to the datasheet for the eb101 Bluetooth Serial Module available on the A7 website. The pin outs are shown here for quick reference.







Name	Pin	Type	Description
RESET	6	CMOS input with weak internal pull-up	Reset if low. Input debounced; must be low for >5ms to cause a reset
UART_CTS	11	CMOS input with weak internal pull-down	UART clear to send, active low
UART_TX	12	CMOS output	UART data output, active high
UART_RTS	13	CMOS output, tristate with internal pull-up	UART request to send, active low
UART_RX	14	CMOS input with weak internal pull-down	UART data input, active high
VDD	15	Supply voltage	Positive supply. Usage is optional if supplying 3.3V to VDD_IO
VDD_IO	16	Supply voltage	Positive supply for UART and I/O ports
STATUS	26	CMOS output	Low when there is an active connection; otherwise high
BREAK	27	CMOS input with weak internal pull-up	Break if low, Input debounced; must be signaled for >5ms to cause a break
SWITCH	28	CMOS input with weak internal pull-up	Signaled if low. Can be used to initiate EasyConnect or factory reset
IND_LED	29	CMOS output	For connection to an indicator LED. A maximum of 8mA may be drawn from this line
HOST_WAKE	30	CMOS output	Active low wake signal. Indicates UART communication to host is imminent
WAKEUP	31	CMOS input with weak internal pull-up	Awake if low. Input debounced; must be low for >5ms to cause a wake
RF (RF Pad version only)	33	Bi-directional analog	Connect to a 50Ω Bluetooth ISM Band antenna

All GND pads are to be grounded. (1, 3, 17, 22, 23, 32, & 34)

All NC pads should be left unconnected. (2, 4, 5, 7, 8, 9, 10, 18, 19, 20, 21, 24, & 25)

A7 eb301 Bluetooth Serial Adapters

The eb301 Bluetooth serial adapters provide not only a board level Bluetooth serial solution; but also a reference design for the eb101 Bluetooth serial module. The complete design package for the eb301 is available on the A7 website including gerbers, schematic, and BOM.

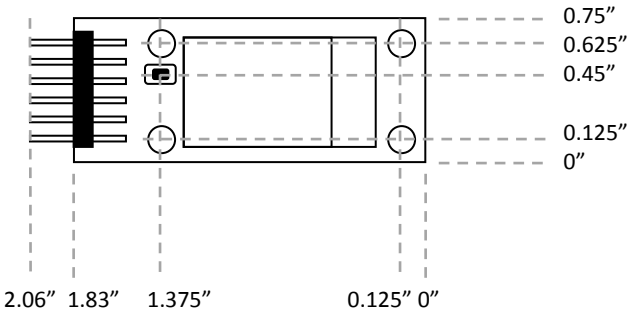
The eb301 is available with the onboard antenna version of the eb101 or the UFL connector version. Both are available with either a 3.3V / 5V (UART) interface or a RS232 level interface. See "UARTs & RS232: A Primer/Refresher" in this guide for more information.

*The **3.3V / 5V UART version** is designed to work with either logic level and, in turn, to be powered accordingly. This version is designed for direct connection to port pins on most microcontrollers.

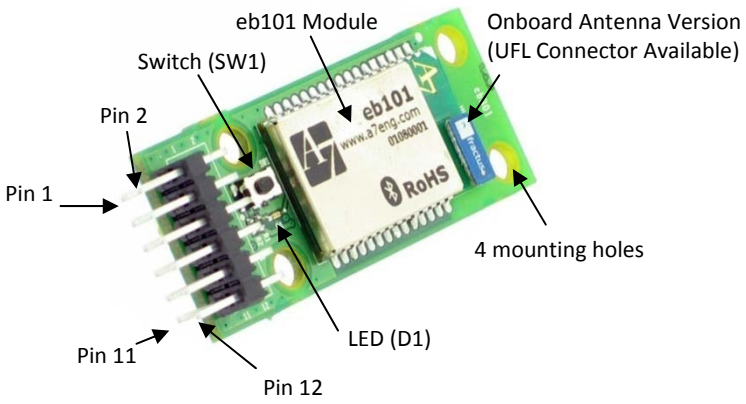
*The **RS232 version** is designed to be powered with 5.5 volts and requires a RS232 level interface typical of devices that have RS232 compatible line driver/receivers.

All versions use the same PWB with version being determined by component population.

The following diagram illustrates the dimensions of the eb301 adapters.



The following picture identifies many of the features of the eb301.



The eb301 features a pin out that provides access to the primary features of the eb101 module.

ⓘ*As the eb301 is designed for embedded systems, the RS232 version of the eb301 would require an adapter to be made to connect to a standard D9 serial port on a personal computer.*

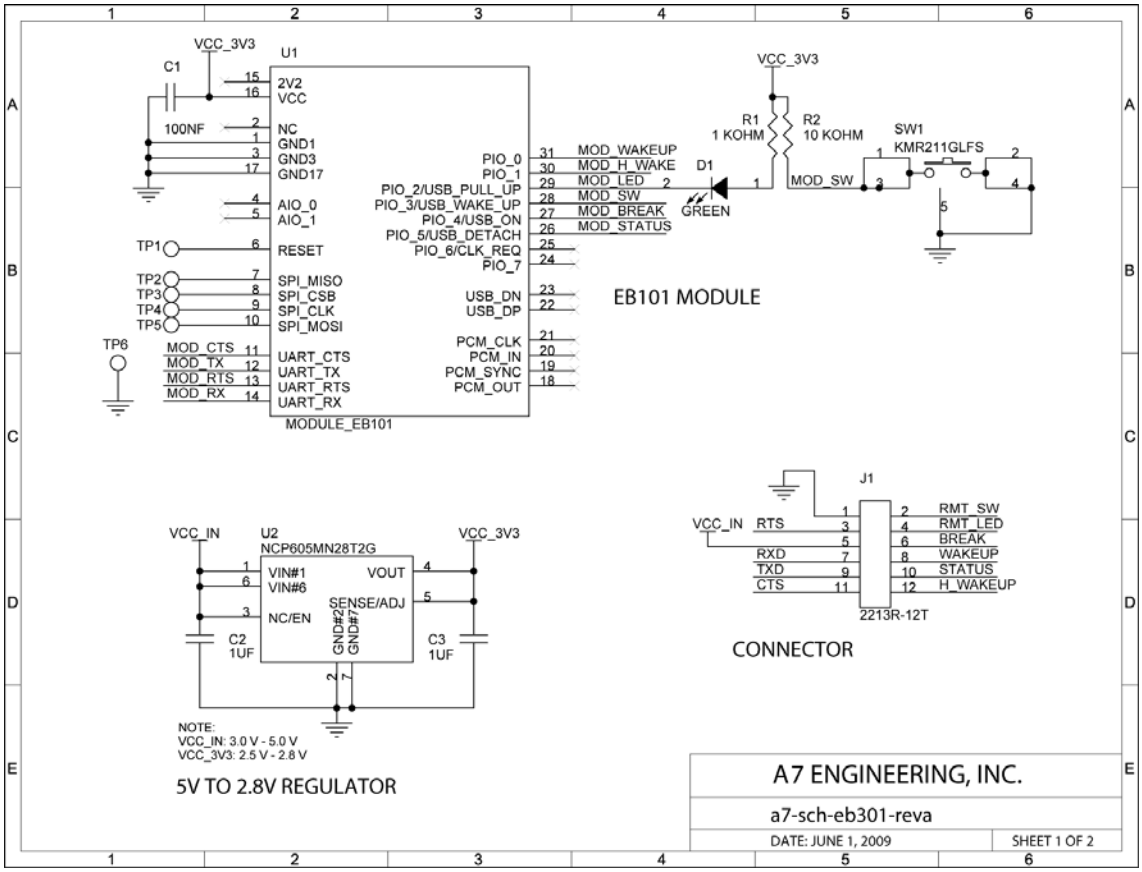
ⓘ*For the 3V/5V version of the eb301, the bottom row (odd pins) is directly compatible with the available FTDI USB to TTL serial cable, part number 1000162, for testing and use on a personal computer. (Black wire, ground, to pin 1).*

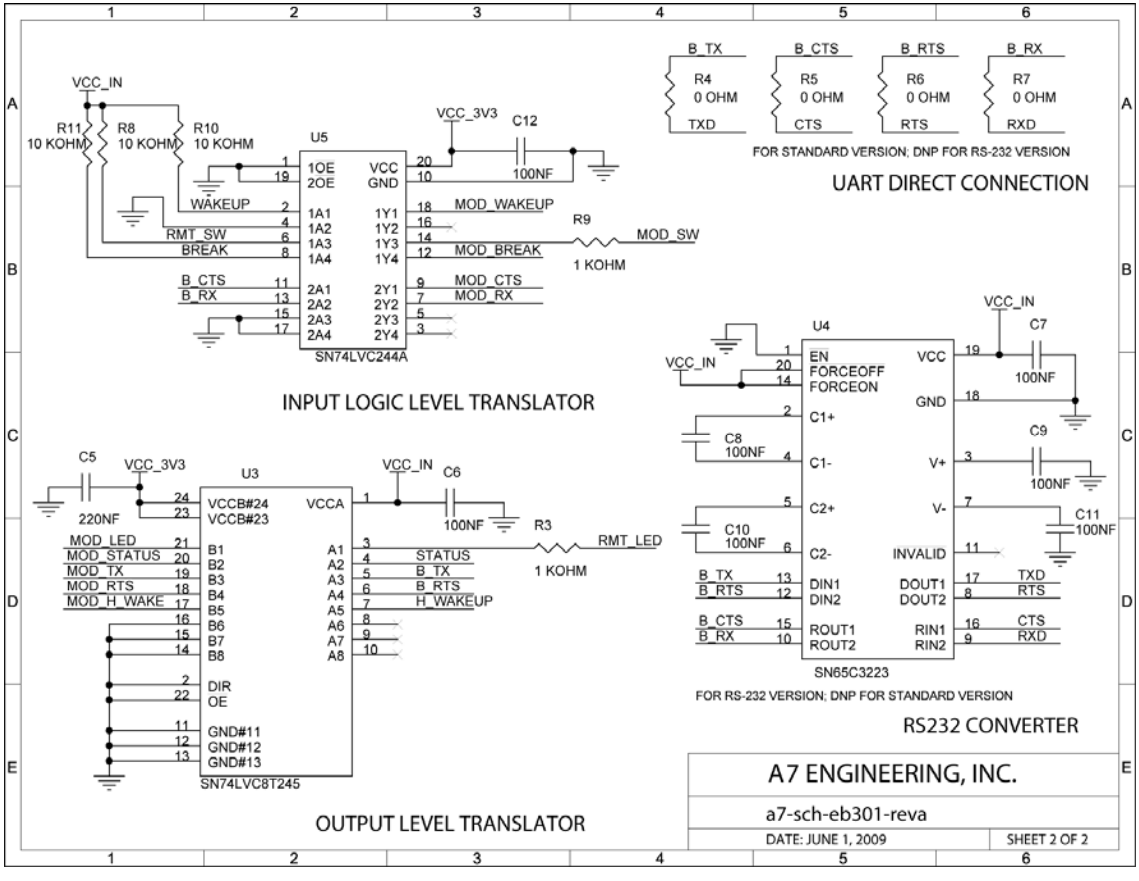
The pin out of the eb301 header is as follows:

```
   ------------------
  | 2  4  6  8 10 12 | <-- 0.1" 2 x 6 Male header
  | 1  3  5  7  9 11 |
 ====================== <- PCB
```

Pin	Direction	Name	Description
Pin 1	<-	GROUND	
Pin 2	<-	RMT_SWITCH	(Optional) Active low input to eb301. Remote switch for EasyConnect and factory reset.
Pin 3	->	RTS	(Optional) Serial ready to send output from eb301.
Pin 4	->	RMT_LED	(Optional) Active low output from eb301. Remote LED for duplicating functionality of onboard LED.
Pin 5	<-	VCC_IN	
Pin 6	<-	BREAK	(Optional) Active low input to eb301. Used to enter command mode and to return to data mode.
Pin 7	<-	RXD	Serial receive input to eb301.
Pin 8	<-	WAKEUP	(Optional) Active low input to eb301.
Pin 9	->	TXD	Serial transmit output from eb301.
Pin 10	->	STATUS	(Optional) Active low output from eb301. Indicates connection active.
Pin 11	<-	CTS	(Optional) Serial clear to send input to eb301.
Pin 12	->	H_WAKEUP	(Optional) Active low output from eb301.

The following pages show the schematic for the eb301 adapter. This schematic may be freely used in part, or in its entirety for designs.


VCC_3V3
U1
C1
100NF
2V2
VCC
NC
GND1
GND3
GND17
AIO_0
AIO_1
RESET
SPI_MISO
SPI_CSB
SPI_CLK
SPI_MOSI
UART_CTS
UART_TX
UART_RTS
UART_RX
MOD_CTS
MOD_TX
MOD_RTS
MOD_RX
TP1
TP2
TP3
TP4
TP5
TP6
PIO_0
PIO_1
PIO_2/USB_PULL_UP
PIO_3/USB_WAKE_UP
PIO_4/USB_ON
PIO_5/USB_DETACH
PIO_6/CLK_REQ
PIO_7
USB_DN
USB_DP
PCM_CLK
PCM_IN
PCM_SYNC
PCM_OUT
MOD_WAKEUP
MOD_H_WAKE
MOD_LED
MOD_SW
MOD_BREAK
MOD_STATUS
MODULE_EB101
EB101 MODULE
R1
1 KOHM
R2
10 KOHM
D1
GREEN
SW1
KMR211GLFS
J1
RMT_SW
RMT_LED
BREAK
WAKEUP
STATUS
H_WAKEUP
2213R-12T
CONNECTOR
U2
NCP605MN28T2G
VIN#1
VIN#6
NC/EN
VOUT
SENSE/ADJ
GND#2
GND#7
C2
1UF
C3
1UF
NOTE:
VCC_IN: 3.0 V - 5.0 V
VCC_3V3: 2.5 V - 2.8 V
5V TO 2.8V REGULATOR
A7 ENGINEERING, INC.
a7-sch-eb301-reva
DATE: JUNE 1, 2009
SHEET 1 OF 2


B_TX
B_CTS
B_RTS
B_RX
R4
0 OHM
TXD
R5
0 OHM
CTS
R6
0 OHM
RTS
R7
0 OHM
RXD
FOR STANDARD VERSION; DNP FOR RS-232 VERSION
UART DIRECT CONNECTION
VCC_IN
R11
10 KOHM
R8
10 KOHM
R10
10 KOHM
U5
VCC_3V3
C12
100NF
1OE
2OE
VCC
GND
WAKEUP
RMT_SW
BREAK
1A1
1A2
1A3
1A4
1Y1
1Y2
1Y3
1Y4
MOD_WAKEUP
R9
MOD_SW
1 KOHM
MOD_BREAK
B_CTS
B_RX
2A1
2A2
2A3
2A4
2Y1
2Y2
2Y3
2Y4
MOD_CTS
MOD_RX
SN74LVC244A
INPUT LOGIC LEVEL TRANSLATOR
U4
VCC_IN
C7
100NF
EN
FORCEOFF
FORCEON
VCC
GND
C1+
C8
100NF
C1-
C9
100NF
V+
C2+
V-
C11
100NF
C10
100NF
C2-
INVALID
B_TX
B_RTS
DIN1
DIN2
DOUT1
DOUT2
TXD
RTS
B_CTS
B_RX
ROUT1
ROUT2
RIN1
RIN2
CTS
RXD
SN65C3223
FOR RS-232 VERSION; DNP FOR STANDARD VERSION
RS232 CONVERTER
C5
VCC_3V3
220NF
U3
VCC_IN
C6
100NF
R3
RMT_LED
1 KOHM
VCCB#24
VCCB#23
VCCA
MOD_LED
MOD_STATUS
MOD_TX
MOD_RTS
MOD_H_WAKE
B1
B2
B3
B4
B5
B6
B7
B8
A1
A2
A3
A4
A5
A6
A7
A8
STATUS
B_TX
B_RTS
H_WAKEUP
DIR
OE
GND#11
GND#12
GND#13
SN74LVC8T245
OUTPUT LEVEL TRANSLATOR
A7 ENGINEERING, INC.
a7-sch-eb301-reva
DATE: JUNE 1, 2009
SHEET 2 OF 2

The Commands

*Note that a lone carriage return cancels any command that is currently in progress with the exception of connect.

Command	Description	Format		
			Parameters	**Parameter Description**
con	Establishes a connection to another Bluetooth device.	con address [profile]<CR>		
			address	Bluetooth address of the remote device.
			profile	Bluetooth profile to connect with. Must be either SPP or DUN.
del	Deletes trusted devices.	del trusted all | address<CR>		
			all	Remove all trusted devices
			address	Bluetooth address of device to delete.
dis	Disconnects the wireless connection.	dis<CR>		
lst	Returns a list of Bluetooth devices.	lst trusted | visible [name] [timeout]<CR>		
			trusted	List all trusted devices.
			visible	List all visible devices.
			name	Include the names of visible devices.
			timeout	Seconds before aborting the list visible attempt (default 15; max 60).
ret	Returns the local device to data mode if there is an active connection.	ret<CR>		
rst	Resets the local device optionally resetting factory defaults.	rst [factory]<CR>		
			factory	Restore factory default settings.
ver	Retrieves the current firmware version.	ver [all]<CR>		
			all	Specifies to return detailed version information.

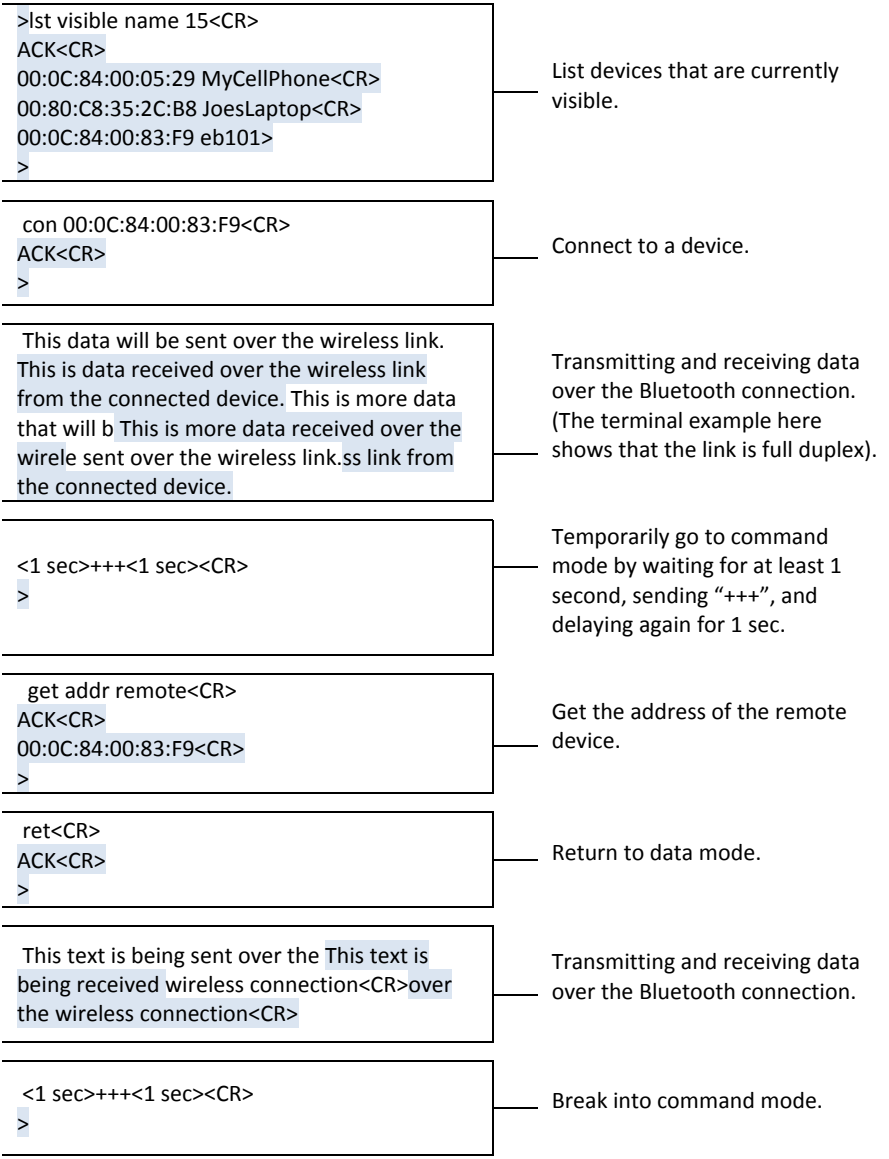
Command	Description	Format		
			Parameters	Parameter Description
get	Retrieves local and remote device settings.	get variable<CR>		
			address local	Returns the local device Bluetooth address.
			address remote	Returns the remote device Bluetooth address.
			connectable	Returns the connectable mode setting; on or off.
			ecbonding	Returns the EasyConnect bonding setting; initiator or responder.
			ecconnection	Returns the EasyConnect connection setting; auto, initiator, or responder.
			encryption	Returns the encryption mode setting; on or off.
			escchar	Returns the current escape sequence character.
			flow	Returns the flow control setting; none or hardware.
			linktimeout	Returns the link timeout setting.
			name local	Returns the local device name.
			name remote	Returns the remote device name.
			parity	Returns the parity setting; odd, even, or none.
			security	Returns the security setting; on or off.
			sleep	Returns the sleep setting; on, off, or idle.
			sniff	Returns the sniff setting; on or off.
			status	Returns the connection status; true or false.
			trustedlist	Returns the trusted list status; on or off.
			txpower	Returns the transmit power setting (1-10)
			visible	Returns the visible mode setting; on or off.

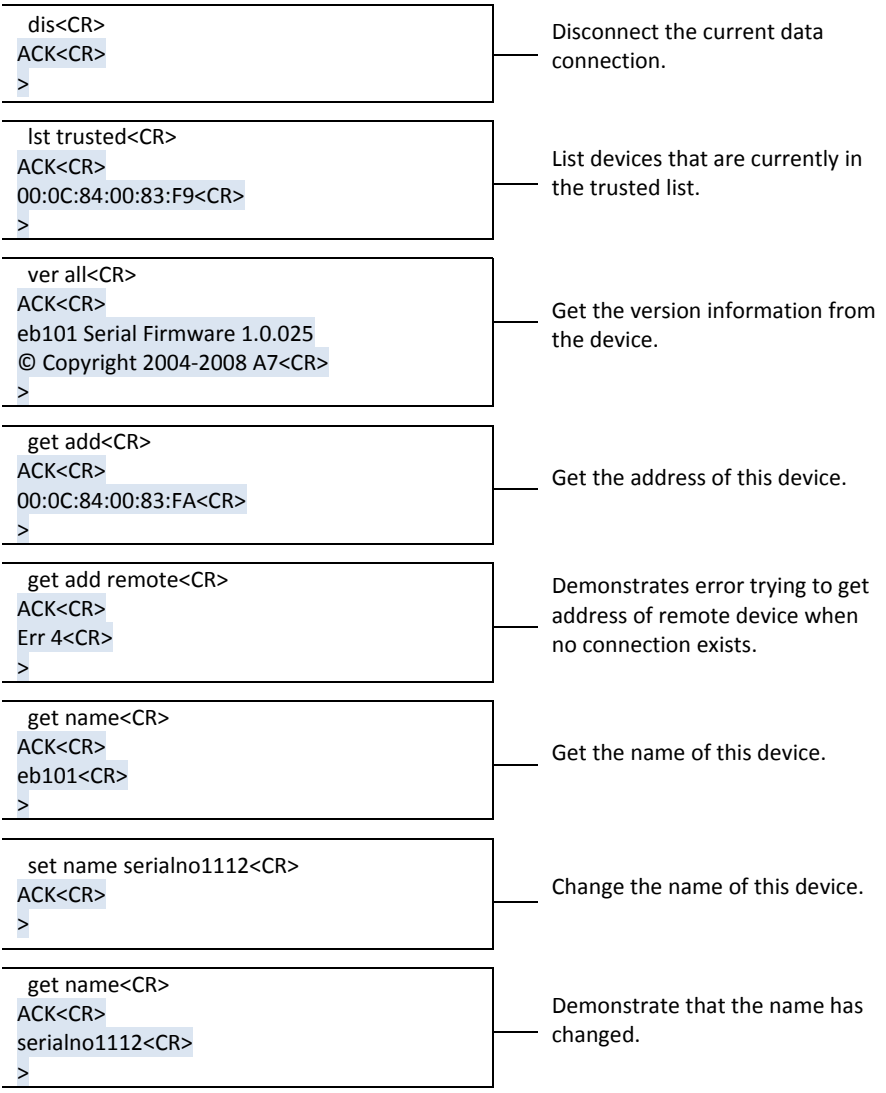
Command	Description	Format		
			Parameters	Parameter Description (Factory defaults in bold)
set	Updates local device settings. Changing these settings will persist until a factory reset is completed. If you have a need to have the firmware with custom factory defaults this can be provided for a nominal fee. Please contact sales@a7eng.com for further information.	set variable value<CR>		
			baud	Baud rate for UART communications using the TXD and RXD lines. Valid values include standard rates from 1200 through 460800. (**9600**)
			connectable	Determines whether a remote Bluetooth device can establish a connection. See Security for more details. Valid values are **on** and off.
			ecbonding	See EasyConnect for detailed description. Valid values are initiator and **responder**.
			ecconnection	See EasyConnect for detailed description. Valid values are **auto**, initiator, and responder.
			encryption	Determines whether transmitted data is encrypted or sent in the clear. The device uses 56-bit encryption to encrypt transmitted data. Valid values are **on** and off.
			escchar	Sets the character used in the soft break command to instruct the local device to break the flow of data to the local host and begin accepting commands. Valid values are any single character. (‘**+**’)
			flow	Determines whether flow control, RTS & CTS, is used for local UART communications. Valid values are **none** and hardware.
			linktimeout	The maximum amount of time, in seconds, that the firmware will try to recover the connection, if lost, before dropping the connection. Valid values are integers from 1 to 40. (**5**)
			name	Sets the local device Bluetooth name. Valid values are strings of up to 32 characters. (**eb101**)
			parity	Determines whether parity is used for local UART communications. Valid values are odd, even and **none**.

			passkey	Passkey that is used when a connection is established with a new device. It is recommended that the passkey is 8 to 16 digits in length. Valid values are strings of up to 16 characters. ("**0000**")
			security	Provides control over whether only trusted devices are permitted to connect. See Security for more details. Valid values are on and **off**.
			sleep	See Power Management Features for detailed description. Valid values are on, **off**, and idle.
			sniff	See Power Management Features for detailed description. Valid values are **on** and off.
			trustedlist	Controls whether devices are added to the trusted device list. See Security for more details. Valid values are **on** and off.
			txpower	Determine the maximum transmitter power the device will use to maintain a connection. This is most often used to limit range as the lower the setting for txpower the less the range for connections will be. Valid values are integers from 1 to **10**.
			visible	Determines whether or not the device is visible to remote Bluetooth devices. Valid values are **on** and off.

Sample Command Scenario

The following diagram illustrates a possible command scenario. The data on the left is the data as it might appear in a terminal application that is directly connected to the A7 eb101 Serial Firmware based device such as the eb301 adapter. The data that is not shaded is the data that is being typed into the terminal to send to the eb101 and the shaded data is the data being received by the terminal from the eb101. The data on the right is a description of what is happening. The row breaks are given for clarity of description only.




dis<CR>
ACK<CR>
>
Disconnect the current data connection.
lst trusted<CR>
ACK<CR>
00:0C:84:00:83:F9<CR>
>
List devices that are currently in the trusted list.
ver all<CR>
ACK<CR>
eb101 Serial Firmware 1.0.025
© Copyright 2004-2008 A7<CR>
>
Get the version information from the device.
get add<CR>
ACK<CR>
00:0C:84:00:83:FA<CR>
>
Get the address of this device.
get add remote<CR>
ACK<CR>
Err 4<CR>
>
Demonstrates error trying to get address of remote device when no connection exists.
get name<CR>
ACK<CR>
eb101<CR>
>
Get the name of this device.
set name serialno1112<CR>
ACK<CR>
>
Change the name of this device.
get name<CR>
ACK<CR>
serialno1112<CR>
>
Demonstrate that the name has changed.

Error Codes

Code	Description
1	Not used.
2	Connection attempt failed. This error occurs when a connection to the remote device could not be established.
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. This error will be generated if the remote device is available, but not responsive.
7	Connection attempt was refused by the remote device.
8	Connection attempt failed due to a server configuration problem. The server channel specified in the SDP record was not registered on the remote device.
9	Not used.
10	Not used.
11	Not used.
12	Command not valid during startup. This error occurs when a command has been issued before the device is fully powered up and initialized.
13	Command not valid while processing. This error occurs when a command is issued while already actively processing a previously issued command.
14	Connection attempt failed. This error occurs when attempting to connect with an invalid Bluetooth address or a device that is not available. It will also occur when the security settings of the remote and local device are incompatible or their passkeys do not match.
15	Connection attempt failed. Unable to connect with the remote device due to a service level connection error.
16	Connection attempt failed. Unable to connect with the remote device because the service level connection is already established.
17	Connection attempt failed. The remote device terminated the connection.
18	Connection attempt failed. An abnormal disconnect occurred while attempting to establish a connection.
19	Connection attempt failed due to an invalid DUN connection request.
20	An unexpected request occurred while attempting to establish a connection.
21	Command processing failed. The command was parsed successfully, but failed during processing.
22	An unexpected message was received.

EasyConnect

This feature provides for very simple pairing of two devices for simple cable replacement scenarios. In order to use the eb101 serial firmware based devices in this scenario the devices must go through the following one time setup procedure. This procedure assumes that the devices are in the factory default state.

1. Power up the eb101 serial firmware based device.
2. Receive the prompt character “>” from the serial connection to the eb101 serial firmware based device.
3. Set the flow control and baud rate using the serial connection to the eb101 serial firmware based device. (Note that once the “set baud” is used to change the baud rate, the host connected to the serial interface will have to set the new baud rate as well.)
4. Remove power from the eb101 serial firmware based device.
5. Power up the eb101 serial firmware based device while activating the SWITCH (EasyConnect) line of the device.
6. Once the IND_LED (indicator) line is asserted the SWITCH (EasyConnect) line must be deactivated.
7. Repeat the above steps 1 – 6 for the second eb101 serial firmware based device.
8. Activate and deactivate (pulse) the SWITCH (EasyConnect) line once more on one of the devices to designate that device as initiator.
9. Wait a moment while the devices complete their pairing.
10. Notice that the IND_LED (indicator) line on both devices begin to toggle at approximately two second intervals. The devices are now paired and may be used for data connection. The connection will automatically be attempted each time the two devices are powered up until the devices are factory reset.

Typically the SWITCH (EasyConnect) line is connected to a push button switch and the IND_LED (indicator) line is connected to an LED so that this process can easily be initiated by a user. See the eb301 design files and documentation for a sample adapter implementation.

Note that when setting up and using EasyConnect; one of the two devices must be the initiator and one must be the responder. The designation for which role the devices play is determined by the ecbonding setting for bonding/pairing setup and by the ecconnection setting once the devices have already been paired. Often the ecconnection state will follow the state of ecbonding; however it may be set independently. The ecbonding setting and the ecconnection setting can be set using the command interface. The defaults are responder and auto, respectively. As shown, in the scenario above, the ecbonding setting may be switched to initiator via the SWITCH line.

Indicator Line (Adapter LED) Pattern

The IND_LED (indicator) line is typically used to drive an LED. The following table provides a description of the various states.

Pattern	Description
Blink Once @ Power On	Command mode; the device can be controlled over the UART connection using the command set.
Blink Twice @ Power On	Device has been previously setup for EasyConnect and will automatically establish a connection.
Solid On	EasyConnect mode setup as responder and is waiting for another device to pair.
Fast Blink (~2Hz)	EasyConnect mode setup as initiator and is trying to find and pair with another device.
Slow Blink (~0.5Hz)	Active Bluetooth connection.
Off	If the device is in command mode then it is idle when the LED is off. If the device is in EasyConnect mode then it is attempting connection. Of course, this could also indicate that no power is applied.

Factory Reset

To reset the eb101 serial firmware to the factory default settings the following procedure may be used.

1. Remove power from the device.
2. Activate the SWITCH line of the device implementing eb101 serial firmware. (Continue to activate this line through the next 3 steps.)
3. Apply power to the device.
4. While continuing to activate the SWITCH line the IND_LED line will be asserted within one second.
5. While continuing to activate the SWITCH line the IND_LED line will be de-asserted.
6. Deactivate the SWITCH line.
7. The device will now be in the factory reset state. In the factory reset state the device will boot to Command Mode.

Example using an eb301 adapter:

1. *Remove power from the eb301 adapter.*
2. *Press the switch, SW1, and hold through the next three steps.*
3. *Apply power to the eb301 adapter.*
4. *While continuing to press the switch, notice that the LED, D1, turns on.*
5. *While continuing to press the switch, notice that the LED turns off.*
6. *Release the switch.*
7. *The eb301 will now be in the factory reset state.*

Security

Bluetooth defines being able to see a device and being able to connect to a device as part of the security model. These features are exposed by the eb101 Serial 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 eb101 serial 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.

Note that it is possible to inhibit devices from becoming trusted with security off by using the "set trustedlist off" command. It is also possible to remove devices from the trusted list, without a factory reset, using the "del trusted" command.

Range

The range that one should expect from an A7 Bluetooth module or adapter depends on a number of factors. The Class 2 spec is for 10 meters or approximately 32 feet; however far more or far less range may be realized due to the actual implementation. Here are some common factors to consider:

- What is the orientation of the device's antenna?
- Is the device's antenna enclosed? If so, consider the material's ability to pass frequencies in the 2.4GHz range.
- Are there objects between the devices that desire a link?
- Are the devices submersed in a liquid? Generally liquids attenuate signals more aggressively at higher frequencies; with Bluetooth being centered at 2.4GHz it is considered fairly high.
- What are the bandwidth requirements for the use case?

The firmware will adjust power dynamically based on requirements for maintaining a connection, up to the maximum set by the txpower setting. The greater the range, the more power the device will use.

Power Management Features

There are a number of features that are employed by the eb101, and in turn the eb301, that allow some level of control over power usage. Overall power usage greatly depends on the use case for each scenario. Understanding the information provided herein will allow for taking advantage of any of the built in features that apply.

Maximum Transmit Power
The firmware allows setting of the maximum level of transmit power the module will use. This is done using the “set txpower” command. The module will use the current setting as its maximum level when ramping power to maintain a robust connection. This is most effective at limiting range and in turn can be seen as a method for limiting power consumption at the expense of range. The maximum power may be set to integer values from 1 to 10, with 10 being maximum transmit power of the module and the factory default.

Power Off
Power may be removed from the module to allow maximum power savings. Of course, this comes at the cost of boot time and reestablishing a connection as necessary; which often may take in excess of 10 seconds.

Deep Sleep Mode
Deep sleep is an aggressive sleep mode which clocks down the CPU on the module to conserve power. Deep sleep may be enabled in two configurations using the “set sleep” command:

1. “set sleep idle” - Deep sleep only when idle; meaning only when the device does not have an active connection.
2. “set sleep on” - Deep sleep when idle or active connection.

Additionally, the “set sleep off” command may be used to disable this feature; which is the factory default. Once enabled, deep sleep will be entered when all inputs are in the inactive (high) state and there is no activity on the UART lines.

Once deep sleep is entered the device will “wake up” upon activation of an input or traffic on the UART lines. The wake input is designed as a convenient mechanism for wakeup before transmitting data. As long as the wake input is active (low) the module will not go back into deep sleep. The module requires 10 milliseconds to wake up upon activation of any input.

While not recommended, it is possible to use UART traffic to wake the device. In order to prevent the module from going back to sleep between UART traffic, the traffic must continue. Since the leading character(s) may be lost, due to the time to wake up, it is important to “prime” the UART. The time to sleep is dynamic and therefore a method of verification should be used to assure robustness of the data being transmitted as required.

Sniff Mode

Sniff mode is a power management feature which applies to active connections and configures a schedule for communications between the connected devices, during idle periods. This saves power by enabling the transmitter to stay off as much as possible and in turn does increase latency of the first few bytes transmitted after an idle period.

Sniff mode may be enabled and disabled using the “set sniff on” and “set sniff off” commands respectively. Optionally, the “set sniff on” command may be followed by parameters as follows:

`set sniff on [idleTimeout][minInterval maxInterval attempt timeout]`

idleTimeout Configures the time, in seconds, that data must not be flowing in either direction before sniff is engaged. The more frequently that a device switches between idle and active transmission of data, the greater the effect idleTimeout will have on power consumption. (Default = 2; Range: 1 to 600)

minInterval Configures the minimum possible value for the negotiated sniff timing. Sniff timing has a significant effect on power consumption. This parameter is in 0.625ms intervals. If this is set to 18 the minimum possible sniff interval will be 18*0.625 = 11.25ms. (Default = 18; Range: 2 to 65534; only even values are valid)

maxInterval Configures the maximum possible value for the negotiated sniff timing. Sniff timing has a significant effect on power consumption. This parameter is in 0.625ms intervals. If this is set to 256 the maximum possible sniff interval will be 256*0.625 = 160ms. (Default = 256; Range: 2 to 65534; only even values are valid)

Note that the maxInterval * 0.625 must be less than the linktimeout setting; see “set linktimeout” for details.

attempt Configures the number of baseband slots the slave will listen for a transmission from the master at the beginning of a sniff interval. This parameter affects the robustness of the sniff timing and typically should not be changed. (Default = 8; Range: 1 to 32767)

timeout Configures the number of additional baseband slots the slave will listen for the sniff interval. This parameter affects the robustness of the sniff timing and typically should not be changed. (Default = 3; Range: 0 to 32767)

Note that *idleTimeout* may be supplied without the remaining parameters; however if any of the remaining parameters are desired, all must be supplied.

Factory Defaults Quick Reference

baud rate:	9600	parity:	none
flow:	none	security:	off
name:	eb101	passkey:	0000
connectable:	on	ecbonding:	responder
ecconnection:	auto	encryption:	on
escchar:	+	linktimeout:	5
sleep:	off	sniff:	on
trustedlist:	on	txpower:	10
visible:	on		

Command Quick Reference

con address [profile]	del trusted all | address
dis	lst trusted | visible [name] [timeout]
ret	rst [factory]
ver [all]	
get address local	get/set visible
get address remote	set baud
get/set connectable	get/set ecbonding
get/set ecconnection	get/set encryption
get/set escchar	get/set flow
get/set linktimeout	get/set name
get name remote	get/set parity
get/set security	get/set sleep
get/set sniff	get status
get/set trustedlist	get/set txpower

Support

If you have a need for further support on these or other A7 products please checkout our website at www.a7eng.com. You can also email us at sales@a7eng.com or support@a7eng.com.

bridging your world)))™