RN-WIFLYCR-UM

WiFly Command Reference, Advanced Features & Applications

RN-WIFLYCR-UM Version 1.0r 9/21/2012

USER MANUAL

Contents

INTRODUCTION

The Roving Networks WiFly radio module is a complete, standalone embedded wireless LAN access device. The device has an on-board TCP/IP stack and applications, and in the simplest configuration only requires four pins (power, TX, RX, and ground). Once you have performed the initial configuration, the device automatically accesses a Wi-Fi network and sends/receives serial data over the UART.

This user manual is applicable to standalone RN-131 and RN-171 modules, as well as Roving Networks hardware that includes these modules. Although there are some differences, the RN-131 and RN-171 modules support the same ASCII command set. The RN-XV device incorporates the RN-171 module; therefore, all RN-171 hardware features apply to the RN-XV. Table 1 compares the RN-131 and RN-171 module features.

Table 1. Comparing the RN-131 & RN-171

Feature	RN-131	RN-171
Output power	18 dBm (fixed)	12 dBm (programmable)
Lowest power	18 dBm	0 dBm (< 100 mA Tx current)
On-board antenna	Yes	No
Accurate sleep timer	Yes (32 kHz)	No (+/- 10% error)
GPIO pins available	10, GPIO4 – 13 (GPIO1 – 3 are not available for use)	14, GPIO1 – 14
Default firmware	**WiFly_GSX-<*version*>.img**	**WiFly_EZX-<*version*>.img**

Refer to the RN-131 and the RN-171 data sheets on the Roving Networks website at http://www.rovingnetworks.com for more details on their hardware differences and for detailed hardware specifications.

FEATURES

- Fully qualified and Wi-Fi certified 2.4-GHz IEEE 802.11 b/g transceiver
- FCC, CE, ICS certified, and RoHS compliant
- Ultra-low power:
  - Intelligent, built-in power management with programmable wakeup
  - Accepts 3.3-V regulated or 2- to 3-V battery with on-board boost regulators
  - IEEE 802.11 power save and roaming functions
  - RN-131: 4 uA sleep, 35 mA Rx, 210 m Tx at 18 dBm (Tx power not configurable)
  - RN-171: 4 uA sleep, 35 mA Rx, 185 mA Tx at 12 dBm (Tx power configurable)
- Antenna options:
  - RN-131: On-board ceramic chip antenna and U.FL connector for external antenna
  - RN-171: RF pad

- Hardware:
    - 8-Mbit flash memory and 128-Kbyte RAM, 2-Kbyte ROM, 2 Kbyte battery-backed memory
    - 10 general-purpose digital I/O pins (RN-131)
    - 14 GPIO pins (RN-171)
    - 8 analog inputs (14 bits, 1.2 V)
    - UART (1 Mbps host data rate) and SPI slave (2 Mbps host data rate) hardware interfaces
    - Real-time clock for wakeup and time stamping/data logging; auto-sleep and auto-wakeup modes
- Network support:
    - Supports ad hoc and infrastructure mode connections
    - Push-button WPS mode for easy network configuration
    - On-board TCP/IP stack
    - Over the air firmware upgrade (FTP) and data file upload support
    - Secure Wi-Fi authentication via WEP-128, WPA-PSK (TKIP), and WPA2-PSK (AES)
    - Configuration over UART or wireless interfaces using simple ASCII commands
    - Built in networking applications: DHCP client, DNS client, ARP, ICMP ping, FTP client, TELNET, HTTP, UDP, and TCP

CONFIGURATION

The WiFly module operates in two modes: data mode and command mode. In data mode, the module can accept incoming connections or initiate outgoing connections. To configure parameters and/or view the current configuration, you must put the module into command mode (also called configuration mode).

Entering Command Mode

By default, the module is in data mode. Sending the escape sequence **$$$** causes the module to enter command mode. You must send **$$$** together quickly with no additional characters before or after. You must not send a carriage return (<cr>) or line feed after the **$$$** to enter command mode. The module replies with **CMD** to indicate it is in command mode. Once in command mode, you can configure the WiFly device using simple ASCII commands; each command ends with a carriage return <cr>. Most valid commands return **AOK**; invalid ones return an **ERR** description. To exit command mode, send **exit** <cr>. The module responds with **EXIT,** indicating that it has exited command mode and entered data mode.

NOTE: There is a 250-ms buffer before and after the **$$$** escape sequence. If characters are sent before or after the escape sequence within this 250-ms interval, the WiFly module treats them as data and passes them over the TCP or UDP socket, and the module will not enter command mode.

You can view various parameters, such as the SSID, channel, IP address, serial port, and other settings, and configure them in command mode. You send commands to the module through the UART or via remotely via telnet. When using the UART interface, the communications settings should match the WiFly module's stored settings. The default is 9,600 baud, 8 bits, no parity, 1 stop bit, and hardware flow control disabled. You can enter command mode locally over the UART interface at any time irrespective of an active TCP connection.

NOTE: Roving Networks suggests using either the TeraTerm (Windows OS) or CoolTerm (Mac OS-X) terminal emulator program.

When the WiFly module powers up, it attempts to auto-associate with the access point stored in its configuration settings. If the module cannot find the access point, it enters auto association mode, and scans and attempts to join a network. This mode may cause the UART to become unresponsive briefly. To avoid configuration problems, the auto-associate feature is disabled when the module is in command mode, making it easy to configure the module. The auto-associate mode turns on when you exit command mode.

NOTE: If your module is running firmware version 2.20 or lower, the auto-associate feature is NOT disabled when the module is in command mode. Therefore, you may lose data sent to the module while it is unassociated, making it difficult enter command mode and configure the module.

You can disable the auto-associate feature using the **set wlan join 0** command. This command prevents the WiFly module from attempting to associate with a network that does not exist. Alternatively, enable ad hoc mode upon power up using the GPIO9 ad hoc/factory reset jumper. If this jumper is high on power up, the module does not associate with a network. While in ad hoc mode, you can configure the network settings.

Remote Configuration Using Ad Hoc Mode

Using ad hoc mode to configure the device eliminates the need for the module to be associated with a network access point. In ad hoc mode, the module creates it's own "on demand" network to which you can connect via your computer as you would with any other network.

To enable ad hoc mode using hardware, set GPIO9 high (3.3 V) at power up. For the RN-134 board, GPIO9 is on pin 1 on the jumper block (J2). For the RN-174 board, GPIO9 is on the J6 connector. Upon power up with GPIO9 high, the WiFly module creates an ad hoc network with the following settings:

SSID:	WiFly-GSX-*XX*, where *XX* is the final two bytes of the devices MAC address
Channel:	1
DHCP:	Off
IP address:	169.254.1.1
Netmask:	255.255.0.0

With the ad hoc jumper in place, these settings override any saved configuration settings.

From your computer, connect to the WiFly-GSX-*XX* network. This open network does not require a pass phrase or pass key. Currently the WiFly module only supports OPEN mode for creating ad hoc networks.

It may take a few minutes for Windows to assign an IP address and connect to the network. You can check your computer's IP address by running the **ipconfig** command in the Command Window. If connected, this command displays your computer's IP address and netmask.

NOTE: The automatically assigned IP address must be on the 169.254.*x.y* subnet, otherwise the WiFly module will not be accessible. If your computer has both wireless and wired interface hardware, you may need to disable the wired LAN interface hardware before connecting to the ad hoc network. If the wired LAN is enabled, the computer may assign an IP address that is not on the same subnet as the WiFly module.

Once connected with an appropriate IP address, telnet into the WiFly module on port 2000 using the following command:

telnet 169.254.1.1 2000

The module issues the response ***HELLO***. You can now enter command mode using the escape sequence **$$$** and configure the module.

In firmware versions 2.28 and higher, you can disable remote configuration, e.g., for security. To disable remote configuration, use bit 4 in the TCP mode register by issuing the command:

set ip tcp-mode 0x10

COMMAND REFERENCE

Roving Networks WiFly modules support a variety of commands for configuration. This section describes these commands in detail and provides examples.

COMMAND SYNTAX

To issue commands to the module, you send a keyword followed by optional parameters. Commands are case sensitive, and you cannot use spaces in parameters. Use a **$** to indicate a space, e.g., **MY NETWORK** should be written as **MY$NETWORK**. Hex input data can be uppercase or lowercase. String text data, such as the SSID, is case sensitive.

You can use shorthand for the parameters. For example, the following commands are equivalent:

- **set uart baudrate 115200**
- **set uart b 115200**
- **set u b 15200**

NOTE: You cannot use shorthand for command keywords. For example, **s uart baudrate 115200** is illegal.

You can type numbers in decimal (e.g., 115200) or hexadecimal. To enter a number in hex, use **0x**<*value*>. For example, the hex value **FF** would be entered as **0xFF**.

COMMAND ORGANIZATION

There are five general command categories, as shown in Table 2.

Table 2. Command Types

Command Type	Description
Set commands	Set commands take effect immediately and are stored to memory when the **save** command is issued.
Get commands	These commands retrieve the stored information and display it.
Status commands	These commands display the interface status, IP status, etc.
Action commands	Use these commands to perform actions such as scanning, connecting, disconnecting, etc.
File I/O commands	Use these commands to upgrade, load and save configuration, delete files, etc.

NOTE: You must save any changes you make using the **save** command or the module will load the previous settings upon reboot or power up.

When the system boots, all configuration data is loaded into RAM variables from the **config** file. The **set** commands only modify the RAM copy of the system variables. In general, the IP, WLAN, and UART settings require you to save and reboot before they take effect because they operate upon power up. For example, you only associate, set the channel, and obtain an IP address once at power up. Most of the other commands, e.g., COMM settings and timers, take effect immediately, allowing you to change parameters on the fly, minimizing power usage, and saving flash re-write cycles.

Once configuration is complete, you must save the settings to store the configuration data, otherwise it will not take effect upon reboot or reset. You can store multiple configurations using the **save** <*filename*> command, and you can load them using the **load** <*filename*> command.

SET COMMANDS

These commands begin with the `set` keyword and include the categories shown in Table 3:

Table 3. Set Commands

Parameter	Description
adhoc	Controls the ad hoc parameters.
broadcast	Controls the broadcast hello/heartbeat UDP message.
comm	Sets the communication and data transfer, timers, and matching characters.
dns	Sets the DNS host and domain.
ftp	Sets the FTP host address and login information.
ip	Specifies the IP settings.
option	Supports optional and infrequently used parameters.
sys	Sets system settings such as sleep and wake timers.
time	Sets the timer server settings.
uart	Specifies the serial port settings such as baud rate and parity.
wlan	Sets the wireless interface settings, such as SSID, channel, and security options.

set adhoc beacon <*value*>

This command sets the ad hoc beacon interval in milliseconds, where <*value*> is a decimal number from 0 to 65,436.

Default: 100

Example: **set adhoc beacon 120** // Beacons are sent every 120 ms

set adhoc probe <*value*>

This command sets the ad hoc probe timeout in seconds, where <*value*> is the number of seconds. The probe timeout is the number of seconds the module waits for probe responses before declaring, "ADHOC is lost," and disabling the network interface.

Default:	5	
Example:	**set adhoc probe 80**	// Sets the ad hoc probe timeout to 80 s

set broadcast address <*address*>

This command sets the address to which the UDP hello/heartbeat message is sent, where <*address*> is an IP address in the form <*value*>.<*value*>.<*value*>.<*value*> with <*value*> being a number between 0 and 255.

Default:	255.255.255.255	
Example:	**set broadcast address 255.255.255.255**	// Sets the broadcast address to 255.255.255.255

set broadcast interval <*mask*>

This command sets the interval at which the hello/heartbeat UDP message is sent and is specified in seconds. The value is a mask that is ANDed with a free running seconds counter; if the result is all 0s, a packet is sent. For example:

- If the interval is 0x1, the module sends one packet every 2 seconds.
- If the interval is 0x2. The module sends two packets every 4 seconds.
- If the interval is 0x3, the module sends one packet every 4 seconds.
- If the interval is 0x6, the module sends two packets every 8 seconds.
- If the interval is 0x7, the module sends one packet every 8 seconds.

The minimum interval value is 1 (every 2 seconds) and the maximum value is 0xff (every 256 seconds). Setting the interval value to zero disables UDP broadcast messages.

Default:	7	
Example:	**set broadcast interval 6**	// Sets the heartbeat UDP message interval to 6 seconds

set broadcast port <*value*>

This commands sets the port to which the UDP hello/heartbeat message is sent, where <*value*> represents the port number.

Default:	55555	
Example:	**set broadcast port 55555**	// Sets the port to which the UDP heartbeat is sent to // 55555

set comm $ <*char*>

This command sets character used to enter command mode to <*char*>. You typically use this setting when **$$$** (the default string used to enter command mode) is a possible data string. You must carefully note the new character. After you save this setting, upon every subsequent reboot the module ignores **$$$** and looks for <*char*><*char*><*char*> to enter command mode.

Default:	$	
Example:	**set comm $ w**	// Sets the string to enter command mode to www

set comm close <*string*>

This command sets the ASCI string that is sent to the local UART when the TCP port is closed, where <*string*> is one or more characters up to a maximum of 32 (32 bytes). If you do not wish to use a string, use a zero (0) as the <*string*> parameter.

Default:	*CLOS*	
Example:	**set comm close *port closed***	// Set the string to *port closed*

set comm open <*string*>

This command sets the ASCI string that is sent to the local UART when the TCP port is opened, where <*string*> is one or more characters up to a maximum of 32 (32 bytes). If you do not wish to use a string, use a zero (0) as the <*string*> parameter.

Default:	*OPEN*	
Example:	**set comm open *port open***	// Set the string to *port open*

set comm remote <*string*>

This command sets the ASCI string that is sent to the remote TCP client when the TCP port is opened, where <*string*> is one or more characters up to a maximum of 32 (32 bytes). If you do not wish to use a string, use a zero (0) as the <*string*> parameter.

Default:	*HELLO*	
Example:	**set comm remote *welcome***	// Set the string to *welcome*

set comm idle <*value*>

This command sets the idle timer value, where <*value*> is a decimal number representing the number of seconds. The idle timer value is the number of seconds during which no data is transmitted or received over TCP before the connection is closed automatically. Setting the timer to 0 (the default) means the module never disconnects when idle.

Default:	0	
Example:	**set comm idle 25**	// Set the idle timer value to 25 s

set comm match <*value*> | <*hex*>

This command sets the match character, where <*value*> is a decimal number from 0 to 127 or a hex number from 0 to 7F. When this configuration option is set, the module sends an IP packet each time the match character appears in the data. You enter <*value*> either as the decimal (e.g., 13) or hex (e.g., 0xd) equivalent of the of the ASCII character. Setting the match character to 0 disables matching.

A match character is one of three available methods you can use to control TCP/IP packet forwarding. The other methods are **set comm size** and **set comm time**. For more information refer to “UART Receiver & RTS/CTS Hardware Flow Control” UART Receiver on page 50.

Default:	0	
Example:	**set comm match 13**	// Set the match character to a carriage return

set comm size <*value*>

This commands sets the flush size in bytes, where <*value*> is a decimal number from 0 to 1,420 (at 9600 baud). When this configuration option is set, the module sends an IP packet each time <*value*> bytes are received. Roving Networks recommends that you set this value as large as possible to maximize TCP/IP performance.

NOTE: To optimize the link, this value is set automatically when the baud rate is set. It is assumed that higher baud rates equate to more data, hence the flush size is increased.

Flush size is one of three available methods you use to control TCP/IP packet forwarding. The other methods are **set comm match** and **set comm time**. For more information refer to “UART Receiver & RTS/CTS Hardware Flow Control” UART Receiver on page 50.

Default:	64	
Example:	**set comm size 1420**	// Set the flush size to 1,420 bytes

set comm time <*value*>

This command sets the flush timer, where <*value*> is a decimal number representing milliseconds. When this configuration option is set, the module sends an IP packet if no additional bytes are received for <*value*> ms. Setting this value to 0 disables forwarding based on the flush timer.

The flush timer is one of three available methods you can use to control TCP/IP packet forwarding. The others are **set comm match** and **set comm size**. For more information refer to “UART Receiver & RTS/CTS Hardware Flow Control” UART Receiver on page 50.

Default:	10	
Example:	**set comm time 20**	// Set the flush timer to 20 ms

set dns address <*address*>

This command sets the IP address of the DNS sever, where <*address*> is an IP address in the form <*value*>.<*value*>.<*value*>.<*value*> with <*value*> being a number between 0 and 255. This address is automatically set when using DHCP; you must set the DNS IP address for static IP or automatic IP modes.

Default:	0.0.0.0	
Example:	**set dns address 169.64.1.1**	// Set the DNS server address to 169.64.1.1

set dns name <*string*>

This command sets the name of the host for TCP/IP connections to <*string*>, where <*string*> is up to 32 characters (32 bytes).

Default:	server1	
Example:	**set dns name roving1**	// Set the DNS host name to roving1

set dns backup <*string*>

This command sets the name of the backup host for TCP/IP connections to <*string*>, where <*string*> is up to 32 characters (32 bytes). The FTP client uses the backup string to download the firmware via the **ftp update** command.

Default:	rn.microchip.com	
Example:	**set dns backup roving2**	// Set the DNS host name to roving2

set ftp addr <*address*>

This command sets the FTP server's IP address of the FTP server, where <*address*> is an IP address in the form <*value*>.<*value*>.<*value*>.<*value*> with <*value*> being a number between 0 and 255.

Default:	0.0.0.0	
Example:	**set ftp addr 66.35.227.3**	// Set the FTP server to 66.35.227.3

set ftp dir <*string*>

This command sets the starting directory on the FTP server, where <*string*> is up to 32 characters. To read/write to subfolders, use the \ character. To indicate the root directory, use a period.

Default:	public	
Example:	**set ftp dir demo**	// Sets the FTP server starting directory to demo
	set ftp dir demo\test	// Sets the FTP server starting directory to demo\test
	set ftp dir .	// Sets the FTP server starting directory to the root // directory

set ftp filename <*filename*>

This command sets the name of the file that is transferred when issuing the **ftp u** command, where <*filename*> is the firmware image. If you specify any file other than the firmware image, the WiFly module downloads the file and issues the **UPDATE FAIL=3** error.

Default:	wifly_GSX-<*version*>.img (RN-131), wifly_EZX-<*version*>.img (RN-171)	
Example:	**set ftp filename my_data**	// Sets the firmware image to be retrieved via FTP as // my_data

set ftp mode <*mask*>

This command sets the ftp mode, where <*mask*> indicates active or passive mode.

Default:	0x0	
Example:	**set ftp mode 0x1**	// Enables active FTP mode

set ftp remote <*value*>

This command sets the FTP server's remote port number, where <*value*> is the port number.

Default:	21	
Example:	**set ftp remote 25**	// Sets the FTP server's remote port to 25

set ftp time <*value*>

The command sets the FTP timeout value, where <*value*> is a decimal number that is five times the number of seconds required. The module uses this timer to close the FTP connection automatically after the specified time.

Default:	200	
Example:	**set ftp timer 40**	// Sets a 5-second timer
	set ftp timer 80	// Sets a 10-second timer

set ftp user <*string*>

This command sets the user name for accessing the FTP server, where <*string*> is up to 16 characters (16 bytes).

Default:	roving	
Example:	**set ftp user my_username**	// Sets the user name to my_username

set ftp pass <*string*>

This command sets the password for accessing the FTP server, where <*string*> is up to 16 characters (16 bytes).

Default:	Pass123	
Example:	**set ftp user my_password**	// Sets the user name to my_password

set ip address <*address*>

This command sets the WiFly module's IP address, where <*address*> is an IP address in the form <*value*>.<*value*>.<*value*>.<*value*> with <*value*> being a number between 0 and 255. If DHCP is turned on, the IP address is assigned and overwritten when the module associates with an access point. IP addresses are "." delimited.

Default:	0.0.0.0	
Example:	**set ip a 10.20.20.1**	// Sets the WiFly module's IP address to 10.20.20.1

set ip backup <*address*>

This command sets a secondary host IP address, where <*address*> is an IP address in the form <*value*>.<*value*>.<*value*>.<*value*> with <*value*> being a number between 0 and 255. If the primary host IP is unreachable, the module attempts to reach the secondary IP address (if set).

Default:	0.0.0.0	
Example:	**set ip a 10.20.20.2**	// Sets the WiFly module's secondary IP address to // 10.20.20.2

set ip dhcp <*value*>

This command enables/disables DHCP mode, where <*value*> is a decimal number shown in Table 4. If you set this parameter, the module requests and sets the IP address, gateway, netmask, and DNS server upon association with an access point. Any previously set IP information is overwritten.

Table 4. DHCP Modes

Mode	Protocol
0	Turns DHCP off. The module uses its stored static IP address.
1	Turns DHCP on. The module attempts to obtain an IP address and gateway from the access point.
2	Enables automatic IP, which is generally used with ad hoc networks.
3	Turns on DHCP cache mode. The module uses a previously set IP address if the lease is not expired (or the lease survives reboot).
4	Enables DHCP server in soft AP mode.

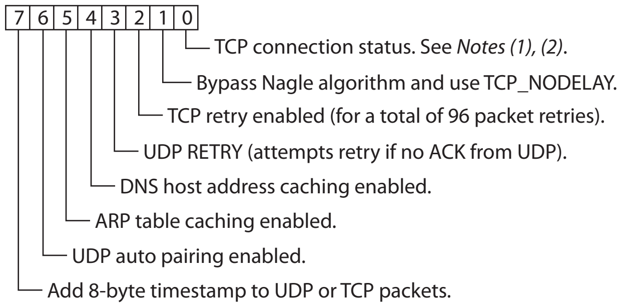
Using DHCP cache mode can reduce the time the module requires to wake from deep sleep, which saves power. The module checks the lease time; if it is not expired, the module uses the previous IP settings. If the lease has expired, the module attempts to associate and uses DHCP to obtain the IP settings. The DHCP cached IP address does not survive a power cycle or reset.

Default:	1	
Example:	**set ip dhcp 0**	// Turns DHCP off

set ip flags <*mask*>

This commands sets the TCP/IP functions, where <*mask*> is a hex number referring to a bit-mapped register. See Figure 1.

Figure 1. set ip flags Command Bit-Mapped Register


7 6 5 4 3 2 1 0
TCP connection status. See *Notes (1), (2)*.
Bypass Nagle algorithm and use TCP_NODELAY.
TCP retry enabled (for a total of 96 packet retries).
UDP RETRY (attempts retry if no ACK from UDP).
DNS host address caching enabled.
ARP table caching enabled.
UDP auto pairing enabled.
Add 8-byte timestamp to UDP or TCP packets.

Notes:

1. If the module loses the link to an associated access point while a TCP connection is active, the TCP connection may hang or be in an inconsistent state. In some cases, the TCP connection will not recover. In firmware version 2.20 and higher, if the module regains the link to the access point within 60 seconds, the TCP connection will survive.
2. In firmware prior to version 2.20, bit 0 specified the TCP copy function.

If bit 0 is set (the default), TCP connections are kept open when the connection to the access point is lost.

If bit 0 is cleared (e.g., by sending **set ip flags 0x6**), if the module loses the access point connection while TCP is connected, the connection is closed.

Default: 0x7

Example: **set ip flags 0x6** // Clear bit 0

set ip gateway <*address*>

This command sets the gateway IP address, where <*address*> is an IP address in the form <*value*>.<*value*>.<*value*>.<*value*> with <*value*> being a number between 0 and 255. If DHCP is turned on, the gateway IP address is assigned and overwritten when the module associates with the access point.

Default: 0.0.0.0

Example: **set ip gateway 169.254.1.1** // Sets the IP gateway to 169.254.1.1

set ip host <*address*>

This command sets the remote host's IP address, where <*address*> is an IP address in the form <*value*>.<*value*>.<*value*>.<*value*> with <*value*> being a number between 0 and 255. You use this command to make connections from the WiFly module to a TCP/IP server with the IP address <*address*>.

Default: 0.0.0.0

Example: **set ip host 137.57.1.1** // Sets the remote host's IP address to 137.57.1.1

set ip localport <*value*>

This command sets the local port number, where <*value*> is a decimal number representing the port.

Default:	2000	
Exampl:	**set ip localport 1025**	// Sets the local port to 1025

set ip netmask <*address*>

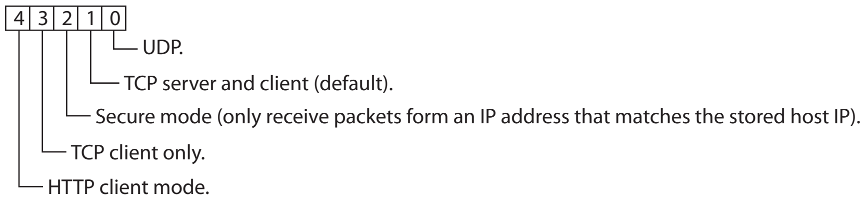
This command sets the network mask, where <*address*> is an IP address in the form <*value*>.<*value*>.<*value*>.<*value*> with <*value*> being a number between 0 and 255. If DHCP is turned on, the netmask is assigned and overwritten when the module associates with the access point.

Default:	255.255.255.0	
Example:	**set ip netmask 255.255.0.0**	// Sets the netmask to 255.255.0.0

set ip protocol <*flag*>

This command sets the IP protocol, where <*flag*> is a bit-mapped register as shown in Figure 2. To be able to connect to the WiFly module over TCP/IP (for example using telnet), you must set bit 2 of the IP protocol register. For the module to accept both TCP and UDP set bits 1 and 2 (value = 3).

Figure 2. set ip protocol Command Bit-Mapped Register


4 3 2 1 0
UDP.
TCP server and client (default).
Secure mode (only receive packets form an IP address that matches the stored host IP).
TCP client only.
HTTP client mode.

Default:	2	
Example:	**set ip protocol** 18	// enables TCP and HTTP client mode

set ip remote <*value*>

This command sets the remote host port number, where <*value*> is a decimal number representing the port.

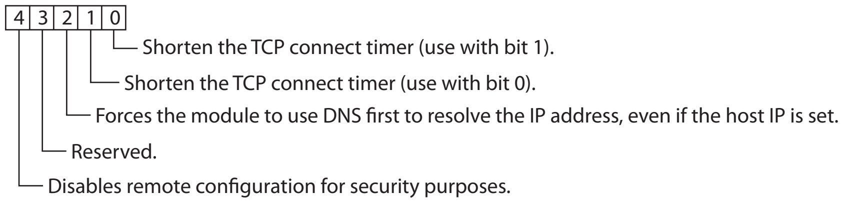
Default:	2000	
Example:	**set IP remote 1025**	// Sets the remote host port to 1025

set ip tcp-mode <*mask*>

This command controls the TCP connect timers, DNS preferences, and remote configuration options. <*mask*> is a hex number referring to a bit-mapped register as shown in Figure 3.

NOTE: The TCPMODE register is available in firmware version 2.27 and higher.

Figure 3. set ip tcp-mode Command Bit-Mapped Register



- Bit 0: Shorten the TCP connect timer (use with bit 1).
- Bit 1: Shorten the TCP connect timer (use with bit 0).
- Bit 2: Forces the module to use DNS first to resolve the IP address, even if the host IP is set.
- Bit 3: Reserved.
- Bit 4: Disables remote configuration for security purposes.

Default:	0x0	
Example:	**set ip tcp-mode 0x4**	// Forces the module to use DNS
	set ip tcp-mode 0x10	// Disables remote configuration

set opt jointmr <*value*>

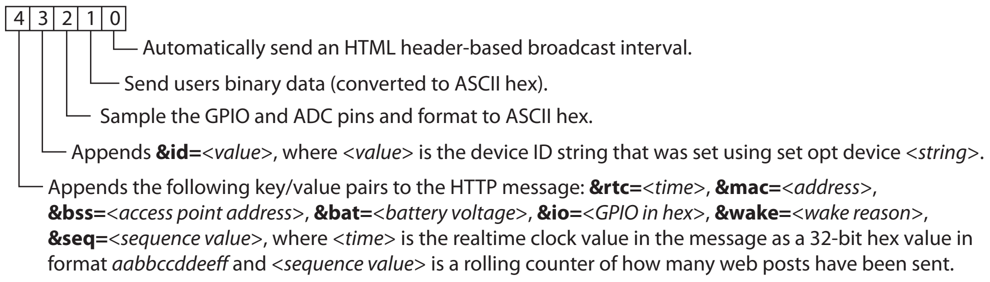
This command sets the join timer, which is the length of time (in ms) the join function waits for the access point to complete the association process. <*value*> is a decimal number representing the number of ms. This timer is also used as the timeout for the WPA handshaking process.

Default:	1000	
Example	**set opt jointmr 1050**	// Sets the join timer to 1,050 ms

set opt format <*flag*>

The command sets the HTTP client/web server information, where <*flag*> is a bit-mapped register as shown in Figure 4. See "Using the HTML Client Feature" on page 66 for more details.

Figure 4. set opt format Command Bit-Mapped Register



- Bit 0: Automatically send an HTML header-based broadcast interval.
- Bit 1: Send users binary data (converted to ASCII hex).
- Bit 2: Sample the GPIO and ADC pins and format to ASCII hex.
- Bit 3: Appends **&id=**<*value*>, where <*value*> is the device ID string that was set using set opt device <*string*>.
- Bit 4: Appends the following key/value pairs to the HTTP message: **&rtc=**<*time*>, **&mac=**<*address*>, **&bss=**<*access point address*>, **&bat=**<*battery voltage*>, **&io=**<*GPIO in hex*>, **&wake=**<*wake reason*>, **&seq=**<*sequence value*>, where <*time*> is the realtime clock value in the message as a 32-bit hex value in format *aabbccddeeff* and <*sequence value*> is a rolling counter of how many web posts have been sent.

Default:	0x00	
Example:	**set opt format 0x7**	// The module sends sensor values

set opt replace <*char*>

This command sets the replacement character you use to indicate spaces in the SSID and pass phrases, where <*char*> is a single character. Each occurrence of the replacement character is changed into a space. Only the WiFly command parser uses this replacement character.

Default:	$ (0x24)	
Example:	**set opt replace %**	// Sets the replacement character to %

set opt deviceid <*string*>

This command sets the configurable device ID, where <*string*> is up to 32 bytes long. You can use <*string*> for serial numbers, a product name, or to show other device information. The module sends the device ID as part of the UDP broadcast hello packet. You can view the device ID's current value with the **get option** or **show deviceid** commands.

Default:	WiFly-GSX	
Example:	**set opt deviceid my_wifly**	// Sets the device ID to my_wifly

set opt password <*string*>

This command sets the TCP connection password, where <*string*> is up to 32 bytes long. This setting provides minimal authentication by requiring any remote device that connects to the module to send and match the challenge <*string*>. When a connection is opened, the module sends the string **PASS?** to the remote host. The remote host must reply with the exact characters that match the stored password in one TCP packet; otherwise, the module closes the connection. To disable the password feature, use **0** (the default).

Default:	"" (no password required)	
Example:	**set opt password my_password**	// Sets the TCP connection password to my_password

set q sensor <*mask*>

This command specifies which sensor pins to sample when sending data using the UDP broadcast packet or the HTTP auto sample function, where <*mask*> is a bit-mapped register.

NOTE: In versions of firmware prior to 2.23, this command is named **set option sensor**.

Default:	0	
Example:	**set q sensor 0xff**	// Enables all sensor inputs

set q power <*value*>

This register automatically turns on the sensor power, where <*value*> is shown in Table 5. This parameter sets an 8-bit register with two 4-bit nibbles. If the top nibble is set, power is applied upon power up and removed upon power down or sleep. If the bottom nibble is set, power is applied when a sampling event occurs such as:

- UDP broadcast
- Automatic web posting of sensor data
- Power is removed immediately after sampling is complete

Table 5. set q power Command Sensor Pin Voltage Settings

Value	Sensor pin voltage
0	Turn off the sensor power.
1	Ground the sensor pin.
2	1.2-V internal regulated reference.
3	VBATT input pin.
4	3.3-V output of on board regulator.

Default: 0

Example:

```
set q power 0x20        // Sets power to 1.2 V automatically upon power up
set q power 0x02        // Sets power to 1.2 V when a sampling event occurs
set q power 0x40        // Sets power to 3.3 V automatically upon power up
set q power 0x04        // Sets power to 3.3 V when a sampling event occurs
```

set sys autoconn <*value*>

This command sets the auto-connect timer in TCP mode, where <*value*> is a decimal number from 0 to 255 as shown in Table 6. Setting this parameter causes the module to connect to the stored remote host periodically as specified by <*value*>.

NOTE: To use the auto-connect timer, you must store the remote host's IP address and port in the WiFly module using the **set ip host** <*address*> and **set ip remote** <*value*> commands.

Table 6. Auto-Connect Timer Settings

Value	Description
0	Disable the auto-connect timer (default).
1	Connect to the stored remote host IMMEDIATELY upon power up or when waking from sleep.
2 - 254	Connect to a stored remote host every <*value*> seconds.
255	Connect to a stored host IMMEDIATELY upon power up or when waking from sleep and go back to sleep IMMEDIATELY as soon as the TCP connection closes.

Default:	0	
Example:	**set sys autoconn 5**	// The module connects to the host every 5 seconds

set sys autosleep <*value*>

This command sets the auto-sleep timer in UDP mode, where <*value*> is a decimal number. If the protocol is set to UDP ONLY, this timer acts as a quick sleep function. The module sleeps <*value*> ms after transmission of the first UDP packet. Setting <*value*> to 0 disables the auto-sleep timer.

Default:	0	
Example:	**set sys autosleep 2**	// Sets the timer to sleep after 2 ms

set sys iofunc <*mask*>

This command sets the I/O port alternate functions, where <*mask*> is a hex number referring to a bit-mapped register. For more details see "Setting GPIO Direction, Alternate Functions & Disabling LEDs" on page 51.

Default:	0x0	
Example:	**set sys iofunc 0x7**	// Disables the WiFly evaluation board LEDs

set sys mask <*mask*>

This command sets the I/O port direction, where <*mask*> is a hex number referring to a bit-mapped register. For more details see “Setting GPIO Direction, Alternate Functions & Disabling LEDs” on page 51.

NOTE: To set the GPIO pins as inputs or outputs instantly, use the **set sys mask 0xABCD 1** command, which does not require a reboot.

Default: 0x20F0 (for RN-131)
0x21F0 (for RN-171)

Example: **set sys mask 0x0** // Sets all pins as inputs

set sys printlvl <*value*>

This command controls the debug print messages printed by the WiFly module on the UART, where <*value*> is one of the values shown in Table 7. Refer to “Setting Debug Print Levels” on page 56 for more information.

Table 7. Debug Print Message Settings

Value	Description
0	Quiet mode. Messages are not printed when the module wakes up or powers up.
1	Print all status messages.
2	Print only critical network access point connection level status, e.g., **Associated!** or **Disconnect from** <*SSID*>.
4	Print the DHCP and IP address status information. After you have verified the module’s configuration, you can turn off this option so that the messages do not interfere with the data.
0x4000	Change the scan format output to an MCU friendly format.
0x10	Enables the UART heartbeat message. See “UART Heartbeat Messages” on page 58 for more details.

Default: 0x1

Example: **set sys printlvl 2** // Sets the debug print messages to only critical network // connection status

set sys output <*mask*> <*mask*>

This command sets the output GPIO pins high or low, where <*mask*> is a hex number referring to a bit-mapped register. The optional <*mask*> sets a subset of the pins.

Default: None

Example: To toggle GPIO8, use the following commands:

```
set sys mask 0x21f0              // Set GPIO8 as output
set sys output 0x0100 0x0100     // Drives GPIO8 high
set sys output 0x0000 0x0100     // Drives GPIO8 low
```

set sys sleep <*value*>

This command sets the sleep timer, where <*value*> is a decimal number. The sleep timer is the time (in seconds) after which the module goes to sleep. This timer is disabled during an open TCP connection. When the TCP connection is closed, the module counts down and puts the module to sleep after <*value*> seconds. Setting the value to 0 disables the sleep timer, and the module will not go to sleep based on this counter.

NOTE: Be sure to set the wake timer before issuing the sleep timer if you are not using an external wake up signal; otherwise, the module will never wake up.

See "System & Auto-Connect Timers" on page 48 for more details on using system timers.

Default: 0

Example: **set sys sleep 5** // Module sleeps 5 s after TCP connection closes

set sys trigger <*flag*> *or* <*mask*>

With this parameter setting, the module wakes from sleep state using the sensor input 0, 1, 2, and 3, where <*flag*> is a decimal number referring to a bit-mapped register as shown in Table 8 and <*mask*> is a hex number. You use either <*flag*> or <*mask*> with this parameter setting. This command sets the sensor input(s) to wake on (0 to 3). Setting <*flag*> to 0 disables wake on sensor inputs.

Table 8. set sys trigger Command Bit-Mapped Register

Bit Position	Description
0	Trigger sensor input 0.
1	Trigger sensor input 1.
2	Trigger sensor input 2.
3	Trigger sensor input 3.
4	Enable WPS function.
5	Enable sleep on GPIO8.

Table 9 describes how you can wake the module using sensor input.

Table 9. Sensor Input Trigger Values

Wake on Sensor Input	Value	Command
0	1	**set sys trigger 1**
1	2	**set sys trigger 2**
2	4	**set sys trigger 4**
3	8	**set sys trigger 8**

NOTE: Setting the system trigger value to **0x10** enables WPS functionality. WPS is disabled by default.

Setting the trigger value to 0x20 (i.e., using <*mask*>) puts the module to sleep when GPIO8 is pulled high. To enable this feature, use the set **sys trigger 0x20** command. This command makes GPIO8 an interrupt pin and puts the module to sleep as soon as it is pulled high, regardless of the module's state; the module goes to sleep even if it is associating with a network or has an open, active TCP connection.

This command is useful for when the module is failing to associate with network because it is out of range (or for any other reason), or if the module must be put to sleep quickly.

NOTE: GPIO8 must be low on power up and stay low until you want to put the module to sleep.

Default: 0x1

Example: **set sys trigger 0x10** // Enable WPS functionality

set sys value <*mask*>

This command sets the default value of the GPIO pins' outputs upon power-up, where <*mask*> is a hex number representing a bit-mapped register. The GPIO pins that are configured as outputs can be driven high or low on power-up or when the module wakes from sleep. The default power-up states can be set ONLY for the GPIO pins that are set as outputs. Setting the value to 1 makes the default power-up state high; setting the value to 0 makes the default power-up state low.

To configure GPIO pins as outputs, use the **set sys mask** <*value*> command.

NOTE: GPIO pins 4, 5, and 6 are used by the firmware to blink the status LEDs. To set the default power up states for these GPIO pins, you must first disable their use by the firmware using the **set sys iofunc 0x7** command.

Default: 0x0

Example: To configure power-up states of GPIO8 (output by default):

set sys value 0x0100 // Sets GPIO8 high upon power-up
set sys value 0x0000 // Sets GPIO8 low upon power-up

set sys wake <*value*>

This command sets the automatic wake timer, where <*value*> is a decimal number representing the number of seconds after which the module wakes from sleep. Setting <*value*> to 0 disables. See "System Timers & Auto-Connect Timers" on page 48 for more details.

Default: 0

Example: **set sys wake 5** // The module wakes after 5 seconds

set time address <*address*>

This command sets the time server address, where <*address*> is an IP address in the form <*value*>.<*value*>.<*value*>.<*value*> with <*value*> being a number between 0 and 255. This command applies to SNTP servers.

Default: 129.6.15.28

Example: **set time address 208.109.78.52** // Sets the time server address as 208.109.78.52

set time port <*value*>

This command sets the time server port number, where <*value*> is a decimal number. 123, the default, is typically the SNTP server port.

Default:	123	
Example:	**set time port 1052**	// Sets the time server port to 1052

set time enable <*value*>

This parameter tells the module how often to fetch the time from the specified SNTP time server, where <*value*> is a decimal number representing minutes. The default (0) disables time fetching. If <*value*> is 1, the module fetches the time only once on power up. If <*value*> is greater than 1, the modules fetches the time every <*value*> minutes.

Default:	0	
Example:	**set time enable 5**	// The module fetches the time every 5 minutes

set time raw <*value*>

This parameter setting allows you to set the RTC raw value from the console, where <*value*> is a decimal number in seconds. The RTC ticks at 32,768 Hz.

Default:	None	
Example:	**set time raw 1**	// Set to 1 second

set uart baud <*value*>

This command sets the UART baud rate, where <*value*> is 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, or 921600.

NOTE: The RN-134 evaluation board's RS-232 interface cannot exceed 230,400 baud.

Default:	9600	
Example:	**set uart baud 19200**	// Sets the baud rate to 19,200 baud

set uart flow <*value*>

This command sets the flow control mode and parity, where <*value*> is a hex number. The setting is in the upper nibble of the hardware flow control setting. The default is flow control disabled with parity set to none/no parity.

NOTE: Once flow control is enabled, it is important to drive the CTS pin properly (i.e., active-low enabled). If CTS is high, the module does NOT send data through the UART and further configuration in command mode is problematic because no response is received.

Default:	0	
Example:	**set uart flow 0x21**	// Even parity with flow control
	set uart flow 0x20	// Even parity without flow
	set uart flow 0x31	// Odd parity with flow control
	set uart flow 0x30	// Odd parity without flow control

set uart instant <*value*>

This command immediately changes the baud rate, where <*value*> is 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, or 921600. This command is useful when testing baud rate settings or when switching the baud rate "on the fly" while connected over TCP via telnet. Using this command does not affect configuration. The module returns the **AOK** response, and then the module exits command mode.

NOTE: In firmware version 2.22 and lower, the module does NOT return an **AOK** over telnet before exiting command mode.

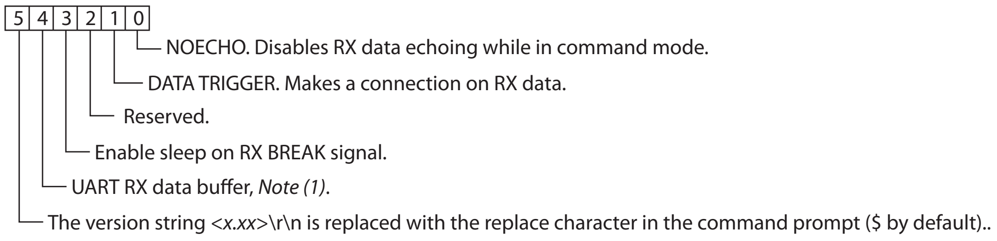
If used in local mode, the baud rate changes and the module sends **AOK** using the new baud rate. If the host switches to the new baud rate immediately, the host may see the **AOK** string at the new baud rate. Depending on the baud rate, it takes at least ten times the bit rate for the module to issue the first character.

Default:	Not applicable	
Example:	**set uart instant 19200**	// Sets the baud rate to 19,200 baud

set uart mode <*mask*>

This command sets the UART mode register, where <*mask*> is a hex number masking a bit-mapped value as shown in Figure 5.

Figure 5. set uart mode Command Bit-Mapped Register


5 4 3 2 1 0
NOECHO. Disables RX data echoing while in command mode.
DATA TRIGGER. Makes a connection on RX data.
Reserved.
Enable sleep on RX BREAK signal.
UART RX data buffer, *Note (1)*.
The version string <x.xx>\r\n is replaced with the replace character in the command prompt ($ by default)..

Notes:

1. With firmware version 2.27 and higher, bit 4's functionality has changed. When a TCP connection is closed, if there is RX data in the UART receiver, it is flushed by default.

When you enable bit 4 using the **set uart mode 0x10** command, the module holds the UART data in the buffer until:

- More characters come in, after which the buffer is flushed.
- No characters come in and a new connection is made, after which the characters are forwarded.

Default:	0	
Example:	**set uart mode 0x10**	// Enable the UART data buffer

set uart raw <*value*>

This command sets a raw UART value, where <*value*> is a decimal number representing the baud rate. You use this command to set non-standard baud rates. The lowest possible baud rate is 2,400.

Using non-standard raw baud rates with hardware flow control can be more useful at speeds as the microcontroller interfaced to the module may be able to better match the UART speed and get better results.

Table 10 shows the supported raw baud rates:

Table 10. Supported Raw Baud Rates

Raw Baud Rate	Comment
458,333	This is 460,800.
500,000	Raw baud rate.
550,000	Raw baud rate.
611,111	Raw baud rate.
687,599	Raw baud rate.
785,714	Raw baud rate.
916,667	This is 921,600.
1,100,000	Raw baud rate.

Default: Not applicable

Example: **set uart raw 7200** // Sets the baud rate to 7,200 baud

set uart tx <*value*>

This command disables or enables the UART's TX pin (GPIO10), where <*value*> is 1 or 0. Disabling the pin (<*value*> = 0) sets GPIO10 as an input with a weak pull down.

NOTE: Firmware version 2.36/2.45 and higher supports parity with the **set uart flow** command.

Default: Not Applicable

Example: **set uart tx 1** // Enable the UART's TX pin

set wlan auth <*value*>

This command sets the authentication mode, where <*value*> is shown in Table 11. You only need to set this parameter if you are using automatic join mode 2, i.e., the **set wlan join 2** command.

NOTE: During association the module interrogates the access point and automatically selects the authentication mode.

The firmware supports he following security modes:

• WEP-64 and WEP-128 (open mode only, NOT shared mode)

• WPA2-PSK (AES only)

• WPA1-PSK (TKIP only)

• WPA-PSK mixed mode (some access points, not all are supported)

Table 11. set wlan auth Command Authentication Modes

Value	Authentication Mode
0	Open (Default)
1	WEP-128
2	WPA1
3	Mixed WPA1 and WPA2-PSK
4	WPA2-PSK
5	Not used
6	Ad hoc mode (join any ad hoc network)
8	WPE-64

Default: 0

Example: **set wlan auth 4** // Use WPA2-PSK authentication

set wlan channel <*value*> <*flag*>

This command sets the WLAN channel, where <*value*> is a decimal number from 1 to 13 representing a fixed channel and <*flag*> is the optional character **i** (meaning immediate). If you set the channel to 0, the modules performs a scan using the SSID for all the channels set in the channel mask. The **i** flag allows you to create a temporary AP mode setup without having to reboot or save the settings. See Example 2.

Default:	0	
Example 1:	**set wlan channel 2**	// Set the WLAN channel to 2
Example 2:	**set wlan channel 1 i**	
	set wlan join 7	
	set ip address 1.2.3.4	
	set ip gateway 1.2.3.4	
	set ip netmask 255.255.255.0	
	set ip dhcp 4	// Use DHCP server
	join <*SSID*>	// Module goes into AP mode

set wlan ext_antenna <*value*>

This commands determines which antenna is active, where <*value*> is 0 (use the chip antenna) or 1 (use the U.FL connector). Only one antenna is active at a time and the module must be power cycled after changing the antenna setting.

NOTE: This command applies only to the RN-131 module; it is not applicable to the RN-171. If you send this parameter to the RN-171, it issues an error message **ERR: Bad Args**.

Default:	0	
Example:	**set wlan ext_antenna 1**	// Use the U.FL antenna

set wlan join <*value*>

This command sets the policy for automatically associating with network access points, where <*value*> is one of the options shown in Table 12. The module uses this policy on powers up, including waking up from the sleep timer.

Table 12. set wlan join Command Options

Value	Policy
0	Manual. Do not try to associate with a network automatically.
1	Try to associate with the access point that matches the stored SSID, passkey, and channel. If the channel is set to 0, the module will scan for the access point. (Default)
2	Associate with ANY access point that has security matching the stored authentication mode. The module ignores the stored SSID and searches for the access point with the strongest signal. You can limit the channels that are searched by setting the channel mask.
3	Reserved.
4	Create an ad hoc network using stored SSID, IP address, and netmask. You MUST set the channel. Unless another ad hoc device can act as DHCP server, set DHCP to 0 (static IP) or use automatic IP. You can use this policy instead of the hardware jumper to create a custom ad hoc network.
7	Create a soft AP network using stored the SSID, IP address, netmask, channel, etc. This mode applies only to firmware versions supporting soft AP mode, not ad hoc mode.

Default:	1 | 0	// For firmware verison 2.36 and lower, auto join is // turned on by default (ad hoc version). For firmware // version 2.45 and higher (soft AP version), auto // join is disabled
Example:	**set wlan join 4**	// Create an ad hoc network

set wlan hide <*value*>

This command hides the WEP key and WPA passphrase, where <*value*> is 0 or 1. If this parameter is set to 0, the pass phrase or passkey is displayed. If you set this parameter to 1, the module shows ****** for these fields when displaying the WLAN settings. To show the pass phrase or passkey, re-enter the key or pass phrase using the **set wlan key** or **set wlan passphrase** command.

Default:	0	
Example:	**set wlan hide 1**	// Hide the pass phrase or passkey

set wlan key <*value*>

This command sets the 128-bit WEP key, where <*value*> is EXACTLY 26 ASCII chars (13 bytes) in hex without the preceding 0x. Hex digits greater than 9 can be either upper or lower case. If you are using WPA or WPA2, enter a pass phrase with the **set wlan passphase** command.

Default: Not applicable

Example: **set wlan key 112233445566778899AABBCCDD** // Sets the passkey

NOTE: The module only supports open mode, 128-bit keys for WEP. WEP-128 shared mode is not supported because it is easily compromised and has been deprecated from the WiFi standards.

set wlan linkmon <*value*>

This command sets the link monitor timeout threshold, where <*value*> is a decimal number representing the number of failed scans before the module declares **AP is Lost** and de-authenticates. If you set this parameter to 1 or more, the module scans once per second for the access point with which it is associated. The module re-attempts the association based on the join policy setting. Roving Networks recommends setting the threshold to 5 attempts, because some access points do not always respond to probes. If you do not set this parameter, there is no way to detect that an access point is no longer present until it becomes available again (if ever).

Default: 0 (disabled)

Example: **set wlan linkmon 5** // Set the number of scan attempts to 5

set wlan mask <*mask*>

This command sets the WLAN channel mask, which is used for scanning channels with auto-join policy 1 or 2, where <*mask*> is a hex number (bit 0 = channel 1). Reducing the number of channels scanned for association increases battery life. This setting is used when the channel is set to 0.

Default: 0x1FFF (all channels)

Example: **set wlan mask 0x0421** // Scans for channels 1, 6, and 11

set wlan phrase <*string*>

This command sets the passphrase for WPA and WPA2 security modes, where <*string*> is 1 to 64 characters (64 bytes). The passphrase is alphanumeric, and is used with the SSID to generate a unique 32-byte pre-shared key (PSK), which is then hashed into a 256-bit number. When you change either the SSID or the passphrase, the module re-calculates and stores the PSK.

If you enter exactly 64 characters, the module assumes that the passphrase is an ASCII hex representation of the 32-byte PSK, and the value is simply stored.

For passphrases that contain spaces, use the replacement character **$** instead of spaces. For example **my pass word** becomes **my$pass$word**. You can change the replacement character using the **set opt replace** command.

Default: rubygirl

Example: **set wlan phrase my_password** // Sets the phrase to my_password

set wlan rate <*value*>

This command sets the wireless data rate, where <*value*> is a value shown in Table 13. Lowering the data rate increases the effective range of the module.

Table 13. set wlan rate Command Options

Value	Wireless Data Rate (Mbits/second)
0	1
1	2
2	5.5
3	11
4 - 7	Invalid
8	6
9	9
10	12
11	18
12	24 (default)
13	36
14	48
15	54

Default: 12

Example: **set wlan rate 13** // Set the data rate to 36 Mbits/second

set wlan ssid <*string*>

This command sets the SSID with which the module associates, where <*string*> is 1 to 32 characters (32 bytes).

NOTE: <*string*> cannot contain spaces. If the SSID has spaces, use the **$** character to indicate the space. For example, **yellow brick road** becomes **yellow$brick$road.** When you use the **get wlan** command to view the SSID, the module properly displays is as **SSID=yellow brick road**.

Default: roving1

Example: **set wlan ssid my_network** // Set the SSID to my_network

set wlan tx <*value*>

This command sets the Wi-Fi transmit power, where <*value*> is a decimal number from 1 to 12 that corresponds to 1 to 12 dBm. The default, 0, corresponds to 12 dB, which is the maximum TX power. Setting the value to 0 or 12 sets the TX power to 12dBm.

NOTE: This command applies only to the RN-171 module; it is not applicable to the RN-131. The transmit power on the RN-131 is fixed to 18 dBm. If you send this parameter to the RN-131, it issues an error message **ERR: Bad Args**.

Default: 0

Example: **set wlan tx 11** // Set the TX power to 11 dBm

GET COMMANDS

These commands begin with the keyword **get** and they display the module's current values. Except where noted, the **get** commands do not have any parameters.

get adhoc

This command displays the ad hoc settings.

Example: **get adhoc** // Show ad hoc settings

get broadcast

This command displays the broadcast UPD address, port, and interval.

Example: **get broadcast** // Show broadcast UDP information

get com

This command displays the communication settings.

Example: **get com** // Show communication settings

get dns

This command displays the DNS settings.

Example: **get dns** // Show the DNS information

get everything

This command displays all of the configuration settings, which is useful for debugging.

Example: **get everything** // Show all configuration settings

get ftp

This command displays the FTP settings.

Example:	**get ftp**	// Show the FTP setttings

get ip <*char*>

This command displays the IP address and port number settings, where <*char*> is the optional parameter **a**. Using <*char*> returns the current IP address.

Example:	**get ip a**	// Display the current IP address

get mac

This command displays the device's MAC address.

Example:	**get mac**	// Show the MAC address

get option

This command displays the optional settings such as the device ID.

Example:	**get option**	// Show the optional settings

get sys

This command displays the system settings, sleep and wake timers, etc.

Example:	**get sys**	// Show the system settings

get time

This command displays the time server UDP address and port number.

Example:	**get time**	// Show the time server information

get wlan

This command displays the SSID, channel, and other WLAN settings.

Example:	**get wlan**	// Show the WLAN settings

get uart

This command displays the UART settings.

Example:	**get uart**	// Show the UART settings

ver

The command displays the firmware version.

Example: **ver** // Show the firmware version

STATUS COMMANDS

These commands begin with the keyword **show** and they return the current values of the system variables. In some cases, e.g., IP addresses, the current values are received from the network and may not match the stored values. Except where noted, the **show** commands do not have any parameters.

show battery

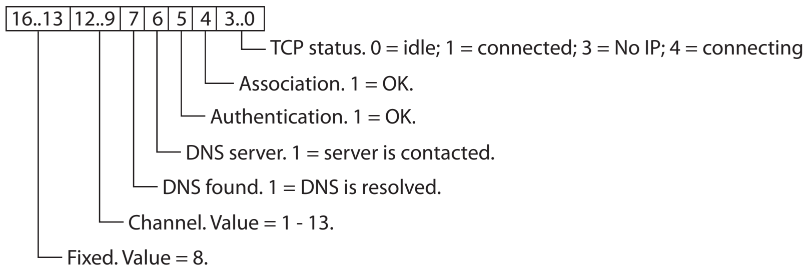
This command displays current battery voltage, and is only applicable to Roving Networks' battery-powered products such as the RN-370 and temperature sensors (ISENSOR-CB).

Example: **show battery** // Show the current battery voltage

show connection

This command displays the connection status in the hex format 8<*XYZ*>, where 8<*XYZ*> is a bit-mapped register providing the information shown in Figure 6.

Figure 6. show connection Command Bit-Mapped Register


16..13
12..9
7
6
5
4
3..0
TCP status. 0 = idle; 1 = connected; 3 = No IP; 4 = connecting
Association. 1 = OK.
Authentication. 1 = OK.
DNS server. 1 = server is contacted.
DNS found. 1 = DNS is resolved.
Channel. Value = 1 - 13.
Fixed. Value = 8.

show io

This command displays the GPIO pins' level status in the hex format 8<*ABC*>. For example, 8103 indicates GPIO 0, 1, and 8 are high.

Example: **show io** // Show the GPIO level status

show net <*char*>

The command displays the current network status, association, authentication, etc., where <*char*> is the optional parameter **n**. Using the **n** parameter displays only the MAC address of the access point with which the module is currently associated.

Example: **show net n** // Show the access point's MAC address

show q <*value*>

This command displays the value of the analog interface pin, where <*value*> is 0 to 7. The A/D reading is 14 bits with a range of 0 to 400 mV (therefore, the resolution is 24 uV). The output is in uV (1,000 millivolts). The module returns a value in the format 8*xxxxx*, where *xxxxx* is the voltage in microvolts sampled on the channel you requested.

NOTE: If a web post or UDP broadcast samples the data, the data is shifted as described in "UDP Broadcast" on page 62.

Example: **show q 0** // Show the voltage on channel 0

show q 0x1<*mask*>

This command displays multiple analog interface values simultaneously, where <*mask*> is a bit-mask representing the channels. For example, to read channels 0, 1, and 7, use the command:

show q 0x183

The module returns 8<*chan0*>, 8<*chan1*>, 8<*chan7*>

NOTE: If a web post or UDP broadcast samples the data, the data is shifted as described in "UDP Broadcast" on page 62.

Example: **show q 0x183** // Show values for channel 0, 1, and 7

show rssi

This command displays the last received signal strength.

Example: **show rssi** // Show signal strength

show stats

This command displays the current statistics, packet RX/TX counters, etc.

Example: **show stats** // Show the statistics

show time

This command displays the number of seconds since the module was last powered up or rebooted.

Example: **show time** // Show seconds since last power up

ACTION COMMANDS

The action commands allow you to enter and exit command mode, join networks, etc. Except where noted, these commands do not have any parameters.

$$$

You use this command to enter command mode. You must type **$$$** together quickly with no additional characters before or after. You must not type a carriage return (<cr>) after the **$$$** to enter command mode. The module replies with **CMD** to indicate it is in command mode. There is a 250-ms buffer before and after the **$$$** escape sequence. If characters are sent before or after the escape sequence within this 250-ms interval, the WiFly module treats them as data and passes them over the TCP or UDP socket, and the module will not enter command mode.

If you want to use different characters to enter command mode (not $$$), you can set the character to use with the **set comm $** <*char*> command.

Example: **$$$** // Enter command mode

close

This command disconnects a TCP connection.

Example: **close** // Close the TCP connection

exit

This command exits command mode. After leaving command mode, the module responds with **EXIT**.

Example: **exit** // Leave command mode

factory RESET

This command loads the factory defaults into the module's RAM and writes the settings to the standard configuration file. You must type the word **RESET** in capital letters. After you type this command, you must reboot the module for the settings to take effect.

Example: **factory RESET** // Reset the configuration settings to the factory defaults

join <*string*>

This commands instructs the WiFly module to join the network indicated by <*string*>. If the network has security enabled, you must first set the pass phrase with the **set wlan pass** command prior to issuing the **join** command.

NOTE: The <*string*> must not contain spaces. If the network SSID contains spaces, use a **$** instead of the space, e.g., **MY$NETWORK** to represent **My Network**.

Default: Not applicable

Example: To join an unsecure network:

join roving1	// Join the network roving1

To join a secure network:

set wlan pass rubygirl	// Set the password to rubygirl
join roving1	// Join the network roving1

join # <*value*>

Use this command to join a network that is shown in the scan list, where <*value*> is the entry number listed for the network in the scan list. If network is security enabled, you must set the passphrase with the **set wlan phrase** command prior to issuing the **join** command.

Example:	**join # 1**	// Join the network indicated by a 1 in the scan list

leave

This command disconnects the module from the access point to which it is currently associated.

Example :	**leave**	// Disconnect from the access point

lites

This command causes the LEDs on the RN-134 evaluation board and RN-370 WiFly serial adapter to blink. Using this command a second time stops the blinking.

Example :	**lites**	// The LEDs begin blinking

lookup <*string*>

This command causes the module to perform a DNS query, where <*string*> is the host name for which to search.

Example :	**lookup roving1**	// Search for the host roving1

open <*address*> <*value*>

This command opens a TCP connection to <*address*>, where <*value*> is the port number and <*address*> is an IP address in the form <*value*>.<*value*>.<*value*>.<*value*> with <*value*> being a number between 0 and 255. If you do not use the <*address*> and <*value*> parameters, the module attempts to connect to the stored remote host IP address and remote port number. <*address*> can also be a DNS host name that the module attempts to resolve.

Default: Stored remote IP address and port

Example: **open 10.20.20.62 2000** // Open a connection to 10.20.20.62 port 2000

ping <*string*> <*value*>

This command pings a remote host, where <*string*> is one of the parameters shown in Table 14 and <*value*> is the number of pings to send. By default, the module sends 1 packet. The optional <*value*> sends < *value*> pings, at 10 per second.

Table 14. ping Command Parameter Options

Option	Description
g	This option pings the gateway. The gateway IP address is loaded if DHCP is turned on; otherwise, you must set it using the **set ip gateway** <*address*> command.
h	This option pings the stored host IP address. You can set the host IP address using the **set ip host** <*address*> command.
i	This option pings a known Internet server, www.neelum.com, by first resolving the URL. This option is useful to demonstrate that DNS is working and that the device has Internet connectivity.
0	This option terminates a previously issued ping command.
<*address*>	Ping a remote host where <*address*> is an IP address in the form <*value*>.<*value*>.<*value*>.<*value*> with <*value*> being a number between 0 and 255.

Default: 1 packet

Example: **ping 10.20.20.12 10** // Ping 10.20.20.12 10 times

reboot

This command forces the module to reboot (similar to a power cycle).

Example: **reboot** // Reboot the module

scan <*value*> <*char*>

This command performs an active probe scan of access points on all 13 channels, where <*value*> is an optional parameter representing the time in ms per channel. <*char*> represents the optional parameter P, which causes the module to perform a passive scan, and list all access points it can see in passive mode.

When you use this command, the module returns the MAC address, signal strength, SSID name, and security mode of the access points it finds. The default scan time is 200 ms/channel or about 3 seconds.

You can also use this command in ad hoc mode.

Default: 200 ms/channel

Example: **scan 30** // Scan for 30 ms/channel

sleep

This command puts the module to sleep. You can wake the module by sending characters over the UART or by using the wake timer.

Example: **sleep** // Put the module to sleep

time

This command sets the real-time clock by synchronizing with the time server specified with the time server (**set time**) parameters. This command sends a UDP time server request packet.

Example: **time** // Set the real-time clock

FILE I/O COMMANDS

You use the file I/O commands to save, load, delete, and update configuration and other files.

del <*string*> <*value*>

This command deletes a file, where <*string*> is the filename and <*value*> is an optional number that overrides the name and uses the sector number displayed by the **ls** command.

Example: **del my_old_config** // Delete the file my_old_config

load <*string*>

This command reads in a new configuration file, where <*string*> is the filename.

Example: **load my_config** // Load the file my_config

ls

This command displays the files in the system.

Example: **ls** // Display the files in the system

save <*string*>

This command saves the your configuration settings to a file, where <*string*> is an optional filename. If you do not specify a filename, the module saves the settings to a file named config (default).

Default:	config	
Example:	**save**	// Saves the configuration settings to the config file
	save my_config	// Saves the settings to the my_config file

boot image <*value*>

This command makes a file represented by <*value*> the new boot image.

Example:	**boot image 55**	// Set the new boot image to a file represented by file // name 55

NOTE: After changing the boot pointer to the new image, you must reboot the module to boot up with the new image. After the module boots up with new image, you must perform a factory reset on the module to initialize all the parameters to the factory default settings. Then you can reinitialize the parameters as required.

ftp update <*string*>

This command deletes the backup image file, retrieves a new image file, and updates the boot pointer to the new image, where <*string*> is the new image file to retrieve. Refer to “Upgrading Firmware Via FTP” on page 71 for more information.

Example:	**ftp update wifly_GSX-2.45.img**	// Retrieve the file wifly_GSX-2.45.img

ADVANCED FEATURES & SETTINGS

This chapter describes the WiFly module's advanced features, including techniques to put the module to sleep, wake up, and methods to open a TCP connection when awake. It also describes the UART flow control, alternative GPIO functions, and the real-time clock.

ACCESS POINT (AP) MODE

Roving Networks WiFly modules support several methods for accessing Wi-Fi networks. In addition to infrastructure mode and ad hoc mode, with firmware version 2.45 the modules support access point (AP) mode. You implement AP mode using special firmware. In the future, AP mode will be released as part of the standard firmware and will replace ad hoc mode.

AP mode provides several advantages over ad hoc mode. In AP mode:

- The module creates a soft AP network to which Android devices (smartphones and tablets) can join. (Android devices do not support ad hoc networking.)
- The module runs a DHCP server and issues IP addresses to seven clients, which is much faster than automatic IP in ad hoc mode.
- The WiFly module will support security in future releases, unlike ad hoc, which is an open mode.
- The module supports routing between clients.

Before enabling AP mode, you must load the firmware that supports it. By default, Roving Networks modules are shipped with firmware version 2.36. You must change the boot image to version 2.45 to use AP mode. You change the module's firmware image using the **boot image** <*value*> command. After you change the boot image, you MUST reset the module back to the factory defaults using the **factory RESET** and **reboot** commands. You can obtain the firmware images from Roving Networks FTP site. See "Upgrading Firmware Via FTP" on page 71 for instructions on how to download and install the firmware.

- **Wifly_EZX-236.img**—Ad hoc mode firmware for RN-171
  Wifly_GSX-236.img—Ad hoc mode firmware for RN-131
- **Wifly_EZX-245.img**—Soft AP mode for RN-171
  Wifly_GSX-245.img—Soft AP mode for RN-131

NOTE: In firmware version 2.36 (ad hoc mode), the auto join feature is enabled to maintain backwards compatibility. In version 2.45 (AP mode), auto join is disabled and you must explicitly enable auto join mode using the **set wlan join 1** command.

The following sections describe how to use AP mode with WiFly products, including configuring the module to act as an AP, enabling AP mode in hardware and software, and sending data to the module from a remote host.

Enabling AP mode

There are two methods for enabling AP mode, hardware and software, as described in the following sections.

Enable in Hardware

To enable AP mode in hardware, hold GPIO9 high at 3.3 V and then reboot (or power cycle) the module. The module boots up in AP mode with the DHCP server enabled.

NOTE: Refer to the documentation for your module on the Support page of the Roving Networks web site at http://rovingnetworks.com/Support_Overview for more details on programming/configuring the module.

Table 15 shows the default AP mode settings.

Table 15. Default AP Mode Settings

Setting	AP Mode Default
SSID	WiFlyAP-*XX*, where *XX* is the last two bytes of the module's MAC address
Channel	1
DHCP server	Enabled
IP address	1.2.3.4
Netmask	255.255.255.0
Gateway	1.2.3.4

When the module boots, other Wi-Fi-enabled devices (PCs, iPhones, iPads, Android tablets, etc.) should be able to see the module when they scan for access points.

NOTE: Roving Networks recommends setting the WiFly module as the gateway when creating a point-to-point network between devices (Wi-Fi network only).

If devices such as smartphones and tablets (iPads, Android tablets, etc.) with a WAN connection associate to the soft AP network, Roving Networks recommends setting the gateway to 0. This setting lets these smartphones route the data from Wi-Fi to the 3G or 4G WAN network.

Enable in Software

You enable AP mode in software using the **set wlan join 7** command. You can customize network settings such as the SSID, channel, and IP address in software to create a custom AP mode. For example, the following commands create a custom AP mode in software:

```
set wlan join 7                   // Enable AP mode
set wlan channel <value>          // Specify the channel to create network
set wlan ssid <string>            // Set up network SSID
set ip dhcp 4                     // Enable DHCP server
set ip address <address>          // Specify the IP address
set ip net <address>              // Specify the subnet mask
set ip gateway <address>          // Specify the gateway
save                              // Store settings
reboot                            // Reboot the module in AP mode
```

After rebooting, the module is in AP mode.

Using AP Mode

The following sections describes how to use AP mode, including connecting to the module, checking for the last device connected over TCP, viewing associated devices, enabling the link monitor, and routing data between clients.

Connect to the Module

Once the module boots up in AP mode, any client device can associate with the network the module is broadcasting. Once associated, the module's DHCP server assigns an IP address to the client device.

The default lease time is 1 day i.e., 86,400 seconds. You can configure the lease time using the **set dhcp lease** <*value*> command, where <*value*> is the time in seconds. To view a list of devices associated with the module, use the **show lease** command. The command output is in the following format with commas delimiting the fields:

IP address assigned	Client MAC address	Remaining lease time (in seconds)	Host name

Figure 7 shows example output from the **show lease** command.

Figure 7. Show Lease Command Example Output

```
<2.42> show lease
1.2.3.10,f0:cb:a1:2b:63:59,153,*
1.2.3.11,00:00:00:00:00:00,0,
1.2.3.12,00:00:00:00:00:00,0,
1.2.3.13,00:00:00:00:00:00,0,
1.2.3.14,00:00:00:00:00:00,0,
1.2.3.15,00:00:00:00:00:00,0,
1.2.3.16,00:00:00:00:00:00,0,
<2.42>
```

NOTE: In AP mode, the module can assign a DHCP lease to 7 clients. However, not all clients report the host name. In this case, the module reports the name as an asterisk (*).

Once a client is associated to the network, it can open a TCP connection to the Roving Networks' module. After successfully opening a TCP connection, the client receives a ***HELLO*** message. The Roving Networks' module prints ***OPEN*** on the UART, indicating an open TCP connection.

Check for the Last Connected Device over TCP

In some cases, it is beneficial to know the IP address of last device that connected to the module over TCP or the last device to which the module connected over TCP. To find this address, use the **show z** command. This command does not survive a power cycle or reboot.

Upon power up, if no device is connected over TCP, the **show z** command returns **0.0.0.0**.

View Associated Devices

To see a list of devices associated with the module, use the **show associated** command. The command output is in the following format with commas delimiting the fields:

Connection number	Host MAC address	Received byte count	Transmitted byte count	Seconds since last packet received

Figure 8 shows example output from the **show associated** command.

Figure 8. Show Associate Command Example Output

```
<2.42> show associated
1,f0:cb:a1:2b:63:59,36868,0,7
2,00:24:8c:31:e5:27,76168,0,2
3,98:4b:4a:6b:e0:0f,1992,0,0
<2.42>
```

You can use the **Seconds since last packet received** output to check for stale connections.

Enable the Link Monitor

AP mode supports a link monitor feature. The link monitor is a timer (in seconds) that checks to see if any packets are received from an associated device. When the timer expires, the AP module de-authenticates the client(s).

You enable the link monitor using the **set wlan link** <*value*>, where <*value*> is the link monitor timer in seconds.

NOTE: Roving Networks recommends that you set the link monitor timer value to at least 300 seconds to avoid de-authenticating clients frequently.

Route Data between Clients

AP mode supports routing between clients. Clients can ping each other via the AP module and can also send data to each other over TCP and UDP.

GPIO4, 5 & 6 Alternative Functions

GPIO4, 5, and 6 have alternative functions in soft AP mode as described in the advanced sections later in this document. You enable the alternative functions using the following commands:

```
set sys iofunc 0x70                  // Enables alternative functions
set wlan linkmon 60                  // Enables link monitor feature required for alternative functions
```

The link monitor feature must be enabled to turn on the alternative functions in soft AP mode only. Table 16 shows the GPIO alternative functions.

Table 16. GPIO Alternative Functions

GPIO	Description
GPIO	Role in alternative function
GPIO4	High when the first client associates, Low when all clients leave the network
GPIO5	WiFly module can drive it high to open a TCP connection to a stored host. When the module drives GPIO5 low, it closes the TCP connection
GPIO6	WiFly module drives it high when a TCP connection is open, low when TCP connection is closed

PUTTING THE MODULE TO SLEEP & WAKING IT

Table 17 describes the methods for putting the module to sleep.

Table 17. Methods for Putting the Module to Sleep

Method	Interface	Description
sleep command	UART	Go into command mode using **$$$** and issue the **sleep** command.
Sleep timer	Internal RTC	The module sleeps based on the **set sys sleep** <*value*> command setting.
Drive GPIO8 high	GPIO8	The module sleeps as soon as GPIO8 is held high (4 μs latency). To enable this feature, use the **set sys trigger 0x20** command setting.

Table 18 describes the methods for waking the module.

Table 18. Methods for Waking the Module

Method	Interface	Description
Sensor input (1.2-V DC only)	Sensor pins	You can wake the module using sensor pins 0 - 3 (1.2-V DC ONLY). Use the **set sys trigger** <*value*> command to enable the sensors.
Rx pin (3.3-V DC only)	RX pin via sensor 0	The RX pin on the RN-134 and the RN-174 evaluation boards is tied to sensor pin 0 via a resistor divider network. Use the **set sys trigger 1** command to wake the module when it receives RX data. **NOTE**: With this method, the module may drop the first UART data byte. A better method is to wake the module using the CTS pin.
CTS pin (3.3-V DC only)	CTS pin via sensor 1	The CTS pin on the RN-134 and the RN-174 evaluation boards is tied to sensor pin 1 via a resistor divider network. Use the **set sys trigger 2** command to wake the module using the CTS pin.
Wake timer	Internal RTC	The wake timer wakes the module based on the **set sys wake** <*value*> command setting.
FORCE AWAKE	FORCE AWAKE pin	An input pulse of at least 31 μs (3.3 V) wakes the module.

When the module wakes up from sleep, it takes time (in milliseconds) to initialize the internal hardware. During this time, any data that is sent to the WiFly module over the UART is not processed. You can monitor signals that indicate the module is ready to accept data, as described in Table 19.

Table 19. Signals Indicating the Module Can Accept Data

Method	Interface	Description
RTS transition	RTS pin	When the WiFly module wakes up, the RTS pin goes high. Once the module is ready, the RTS pin is driven low. You can monitor this pin with a microcontroller.
Monitor GPIO4	Alternative GPIO functions	Set the alternative functions for GPIO4, GPIO5, and GPIO6 (see “Setting the Alternate GPIO Functions” on page 53). When the module wakes up and connects to an access point, GPIO4 goes high, indicating the module is ready to receive data over the UART. A microcontroller can monitor GPIO4.
Sensor power	Sensor power pin	You can configure the module to output Vbat, or 3.3 V or 1.2 V on the sensor power pin when it wakes from sleep, indicating it is ready to accept data.

After the module wakes, you can open a TCP connection to a remote host in a number of ways, as described in Table 20. You set the remote host using the following commands:

set ip host <*address*> *OR* **set dns name** <*string*>	// Sets the host’s IP address OR URL
set ip remote <value>	// Sets the port number on which the host is listening
save	// Save the settings in the configuration file
reboot	// Reboot the module so that the settings take effect

Table 20. Methods of Connecting to a Remote Host

Method	Type	Description
Auto connect	Internal RTC timer	Connect to the host at specific time intervals based upon the **set sys autoconn** <*value*> command setting.
Open	UART	In command mode, issue the **open** command.
Connect on UART data	UART mode 2	This mode is designed for the HTML client feature. Use the **set uart mode 2** command to connect the to host automatically when UART data is received.
GPIO5	Alternative GPIO functions	Set the alternative functions for GPIO4, GPIO5, and GPIO6 (see “Setting the Alternate GPIO Functions” on page 53). Set GPIO5 high to trigger a TCP connection and low to disconnect.

SYSTEM & AUTO-CONNECT TIMERS

The WiFly module uses a real-time clock (RTC) to generate timers. The RTC is active even when the module is asleep, allowing you to put the module to sleep and wake it based on timer intervals. The module has the following timers:

- *Sleep timer*—Used to put the module to sleep. It is a 32-bit number, which corresponds to a maximum 1.19 million waking hours. The sleep timer is set with the **set sys sleep** <*value*> command, where <*value*> is a decimal number representing seconds.
- *Wake timer*—Used to wake the module. It is a 22-bit number, which corresponds to a maximum sleep time of 1,165 hours. The wake timer is set with the **set sys wake** <*value*> command, where <*value*> is a decimal number representing seconds.
- *Auto-connect timer*—Used to open a TCP connection automatically.
- *Idle timer*—Used to close a TCP connection automatically.

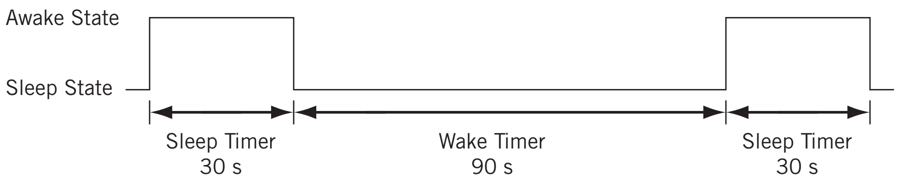
The sleep and wake timers are responsible for putting the module to sleep and waking it up. If the sleep timer is enabled, the module automatically goes into deep-sleep, low-power mode once the timer counts down to 0. The sleep timer is disabled if the module has an IP connection or is in command mode.

For example, if you want the module to wake up, join a network, and be available to accept TCP connections for 30 seconds every 2 minutes you would set the timers as shown in the following example:

```
set wlan ssid my_net                    // Set the host name
set wlan passphrase my_pass             // Set the passphrase
set sys sleep 30                        // The module sleeps after being awake for 30 s
set sys wake 90                         // The module wakes after sleeping for 90 s
save                                    // Save the settings
reboot                                  // Reboot
```

Figure 9 shows the transitions between the sleep and awake state based on the sleep and wake timer settings in the previous example.

Figure 9. Sleep & Awake State Transitions



UDP Sleep & Connection Timers

In UDP only protocol mode (set with the **set ip proto 1** command), the autoconn timer is used as an auto-sleep timer. When the module begins to transmit the first UDP data packet, this timer begins counting down. When it reaches 0, the module sleeps.

You set the UDP auto-sleep timer using two commands, **set sys autosleep** and **set comm timer**. The timer interval is a product of the **autosleep** value and the comm flush timer (in ms). The timer is decremented every "product" milliseconds.

For example, if you want a UDP sleep timer of 40 ms, use the following commands:

set sys autosleep 4	// Set the auto-sleep value to 4
set comm timer 10	// Set the comm timer to 10 ms (default value)

The resulting UDP sleep timer is 4 x 10 ms or 40 ms. You could also set **autosleep** = 2 and **comm timer** = 20 ms to achieve the same effect.

Roving Networks recommends using a minimum value of 2 (when the default flush time is 10 ms) to ensure that the UDP packet is transmitted. For larger packets, you should increase the value.

TCP Connection Timers

The TCP connection timers control when the module opens or closes a socket.

Opening a TCP Connection

In TCP client mode, the auto-conn timer controls the establishment of a socket connection. When set, the device periodically attempts to establish a connection when the timer expires.

The **set sys autoconn** <*value*> command causes the module to connect to the host periodically. The timer <*value*> determines how often to connect to the stored remote host. If set to 1, the module makes one attempt to auto connect upon power up. If set to 2 or higher, auto connect re-opens the connection after the connection is closed. The default, 0, disables the timer.

NOTE: You must specify the remote host's IP address and port number in the module's configuration file for the auto-connect timer to work.

Closing the TCP Connection

The module supports a disconnect timer in both TCP client and server mode (default mode). You can use this timer to close a TCP connection automatically after a specified number of seconds of no transmit or receive data. To set the disconnect timer, use the **set comm idle** <*value*> command, where <*value*> is the number of seconds. The default comm idle timer value is 0, which means the module never disconnects when idle.

For example, to close the TCP connection after 5 seconds of inactivity, use the **set comm idle 5** command.

WAKE ON SENSOR INPUT

Four sensor inputs (0 to 3) wake the module from sleep. These pins have a small current source that is activated in sleep mode. This source is approximately 100 nA, and causes the input to float up to about 1.2-V DC. If, for example, SENSE1 is enabled, pulling the SENSE1 pin to ground wakes the device.

To enable the sensors to wake the module, use the command **set sys trigger** <*mask*>, where <*mask*> is a bit-mapped setting of each sensor. For example, to wake the module using sensor pin 2, use **set sys trig 4**. Setting the trigger value to 0 disables all sensor pins.

Table 21 describes the values to wake the module using individual sensor inputs.

Table 21. Sensor Input Values

Wake on Sensor Input	Value	Command
0	1	**set sys trigger 1**
1	2	**set sys trigger 2**
2	4	**set sys trigger 4**
3	8	**set sys trigger 8**

WARNING: The voltage on any sensor input CANNOT exceed 1.2-V DC or the module will be permanently damaged.

The sensor inputs are rated 1.2-V DC, maximum. You must use a resistor divider when driving a sensor pin from the other 3-V pins such as RX. You should use a resistor divider network with a minimum of 24 K in series and 10 K to ground from the UART RX or CTS pin.

An open-drain FET is an appropriate device to tie to the sensor pin as the threshold is about 500 mV. You can use additional pull-up to 1.2-V DC if the circuit has an impedance (due to leakage current) of less than 5 Mohms (500 mv/100 nA). Leave unused sensor pins disconnected.

WAKE ON UART ACTIVITY

When the module is in sleep mode, the UART is disabled. However, the module can wake on UART activity by connecting the sensor pins to the RX data or CTS pin (using the appropriate divider resistors as described in "Wake on Sensor Input" on page 49).

The RN-134 and the RN-174 evaluation boards have a built in resistor divider connecting SENSE0 and SENSE1 to RXD and CTS, respectively. This setup allows wake on RX and CTS using a 3.3-V signal.

NOTE: Do not apply 3.3 V directly to SENSE0 and SENSE1; the voltage on any sensor input CANNOT exceed 1.2-V DC or the module will be permanently damaged.

To enable wake on RXD, use the **set sys trig 1** command.

It should be noted that the first (or possibly multiple) byte sent to the module will likely be lost; therefore, you should take care to send a preamble byte to wake the module before sending valid data bytes. Alternatively, use the CTS input to wake the module and wait until it is ready to accept data. To enable this setting, use the **set sys trig 2** command.

NOTE: On the RN-134 evaluation board revision 2, the resistor pack connecting the RX and CTS signals is not correctly connected to the sensors. To wake on UART RX, place a jumper from pin 3 on the evaluation board header to pin 2 on the sensor header. To wake on UART CTS, place a jumper from pin 10 on the evaluation board header to pin 3 on the sensor header.

UART Receiver & RTS/CTS Hardware Flow Control

The UART receive buffer is approximately 1,500 bytes. At lower baud rates (less than 115 K), the system can send data over TCP/IP without flow control.

Depending on the frequency and quantity of the data being sent, the **comm** parameters optimize Wi-Fi performance by specifying when the system sends IP packets. To minimize latency and TCP/IP overhead, use the flush size or match character to send data in a single IP packet. In most cases, you should set the flush timer to a large number to avoid fragmentation. For high throughput, increase the UART baud rate, set the flush size to 1,460, and set the flush timer to a large value so that full IP packets are sent.

You can control packet forwarding in the following ways:

- **set comm match** <*value*> sets the value of the packet terminator. Each time the module sees the match character it sends an IP packet. For example, **set comm match 0xd** forwards a packet when the module sees a **0xd** hex character.
- **set comm size** <*value*> sets the flush size, where <*value*> is the number of bytes received before forwarding. The maximum is 1,460 bytes, which is the size of a single Ethernet frame.
- **set comm time** <*value*> sets the flush timer, which is used to flush any partial data sitting the RX buffer if no additional data is received for <*value*> ms. For example the **set comm time 1000** command causes the module to wait for 1 second after no data was sent.

If the module will be sending more than a few hundred thousand bytes in a single transaction, you should enable hardware flow control. Your hardware must actively monitor the CTS pin. Flow control is not enabled by default; you set it with the **set uart flow 1** command.

It is possible to operate higher baud rates (i.e., greater than 115 K) without flow control if the packets are uniform and you use an application protocol to ensure that the packet data is delivered on the remote side before the next packet is sent. However, given the uncertainty of packet delays in a TCP/IP network and the affects of interference and retries inherent in wireless networks, flow control is typically required whenever large, contiguous quantities of data are being written to the UART to guarantee no data is lost.

Setting GPIO Direction, Alternate Functions & Disabling LEDs

You can control the GPIO pin direction with the GPIO mask using the **set sys mask** <*value*> command, where <*value*> is entered as a hex number. The hex number represents a bitmask that controls each pin, where 1 = output and 0 = input. For example:

```
set sys mask 0x0                    // Sets all pins as inputs
set sys mask 0xc0                   // Set only GPIO6 and GPIO7
```

If you only need to set one bit in the mask, you need to read, mask, and set the value. Otherwise, you will overwrite any previous GPIO settings.

The default mask for the RN-131 module is **0x20f0**, which sets GPIO13, GPIO7, GPIO6, GPIO5, and GPIO4 as outputs.

The default mask for the RN-171 module is **0x21f0**, which corresponds to the following settings:

- GPIO0 - 3 are used internally on the module.
- GPIO4 - 6 are LEDs.
- GPIO 9 is reserved as the ARM factory reset/ad hoc mode (read at power up) and otherwise general-purpose input detect pin.
- GPIO10 - 11 are the UART RX and TX pins; TX does not need to be masked as an output.
- GPIO12 is CTS (input), if used.
- GPIO13 is RTS (output), if used.

NOTE: To set the GPIO pins as inputs or outputs instantly, use the **set sys mask 0xABCD 1** command, which does not require a reboot.

The RN-134 evaluation board's LEDs are connected to GPIO4 - 6. To disable the LEDs, enable the alternative functions of the LEDs (use the **set sys iofunc 0x7** command).

NOTE: You can turn off the yellow, red, or green LEDs. The RN-134 board's blue LED is the power indicator and cannot be turned off.

The RN-174 evaluation board's blue LED is connected to GPIO7, which is output by default. The board does not drive this because GPIO7's default power-up state is low.

The **get sys** command shows the setting of the GPIO mask as shown in the following example:

```
<2.21> get sys
SleepTmr=……
IoFunc=0x0
IoMask=0x21f0
```

Figure 10 shows the bits corresponding to the GPIO pins and Table 22 shows the GPIO pin usage, their default state, and functionality.

Figure 10. GPIO Pin Bitmask

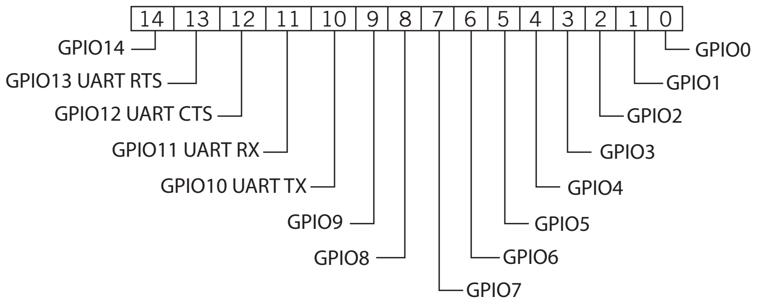


Table 22. GPIO Pin Usage, Default State & Functionality

Bit	Signal Name	RN-131 Default State	RN-171 Default State	Default Function
0	GPIO0	N/A	N/A	-
1	GPIO1	N/A	Input	Unused.
2	GPIO2	N/A	Input	Unused.
3	GPIO3	N/A	Input	Unused.
4	GPIO4	Output	Output	Green LED.
5	GPIO5	Output	Output	Yellow LED.
6	GPIO6	Output	Output	Red LED.
7	GPIO7	Output	Output	Blue LED.
8	GPIO8	Input	Output	Unused.
9	GPIO9	Input	Input	Ad hoc mode and factory reset.
10	GPIO10	Output	Output	UART TX.
11	GPIO11	Input	Input	UART RX.
12	GPIO12	Input	Input	Throttles the transmitter if hardware flow control is enabled. Driving this pin low enables transmitter; driving this pin high disables it.
13	GPIO13	Output	Output	This pin goes high on power up and goes low when the system is ready. If hardware flow control is enabled, this pin toggles to high to indicate the RX buffer is full.
14	GPIO14	N/A	Input	-

NOTE: On the Wi-Fi serial adapter (RN-370) and the RN-174 evaluation board, the blue LED is connected to GPIO7. The blue LED is NOT connected to GPIO7 on the RN-134 board. It is not possible to power off the blue LED on the RN-134 board because it is connected directly to power.

Setting the Alternate GPIO Functions

The GPIO4, 5, and 6 default function is to control the LEDs. You can override the default to allow user programmable I/O or alternate I/O functionality by using the **set sys iofunc** <*mask*> command, where <*mask*> is entered as a hex number. The hex value represents a bitmask that controls each bit in the <*mask*> and represents a particular GPIO pin. If a bit is 0, then the corresponding GPIO pin is driven/read by the firmware per the default function. The I/O function <*mask*> is encoded as shown in Table 23.

Table 23. GPIO Pin Alternate Function Bitmask

Bit	Signal Name	Direction	Function
0	GPIO4	Output	Disable the LED function so the I/O can be used as a GPIO pin.
1	GPIO5	Output	Disable the LED function so the I/O can be used as a GPIO pin.
2	GPIO6	Output	Disable the LED function so the I/O can be used as a GPIO pin.
3	Unused	-	-
4	GPIO4	Output	This pin goes high after the module has associated/authenticated and has an IP address.
5	GPIO5	Input	Set this pin high to trigger a TCP connection and low to disconnect.
6	GPIO6	Output	This pin goes high when the module is connected over TCP and low when disconnected.

NOTE: Bits 0 - 3 are mutually exclusive with bits 4 – 6, i.e., **0x77** is an illegal value.

If you disable the LEDs using bits 0, 1, and 2, you can then use the **show i** command to read these GPIO pins. For example, the **show i** command might return **Port=30**.

To use the alternate LEDs functions, use the following commands:

```
set sys iofunc 0x70                 // Enable alternate function for GPIO4 - 6
save                                // Store configuration
reboot                              // Reboot the module
```

Controlling Connections with GPIO Pins

In embedded applications it is useful to monitor and control the status of the TCP/IP connection. To monitor and control the module's connection status, enable the alternate function of GPIO4 - 6. Using the alternate function for these GPIO pins, the module connects to the stored remote host IP address and port when GPIO5 is driven high and disconnects when driven low. You can monitor the TCP/IP connection status by reading GPIO6; it is high when connected and low when not connected.

To configure the module to connect using GPIO5 and GPIO6, use the following commands:

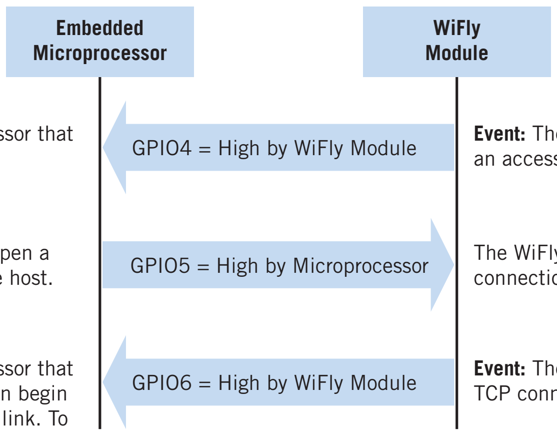
```
set ip host <address>               // Set the IP address of the remote host
set ip remote <value>               // Set the IP port of the remote host
set sys iofunc 0x70                 // Set alternate function for GPIO4 - 6
save                                // Store configuration
reboot                              // Reboot the module
```

After executing these commands, run your application or other software on the remote host that opens and listens on the specified port. Then, connect GPIO5 to your embedded processor or other control signal. When GPIO5 is driven high, the module attempts to connect. When GPI05 is driven low, the connection is closed.

NOTE: Do not to drive the GPIO pin with more than 3.3-V DC or permanent damage to the module will occur.

If the connection to the remote host is successful, GPIO6 goes high. If you have set the **COMM OPEN** and **REMOTE** strings, the UART displays ***OPEN*** and the remote host displays ***HELLO***. Figure 11 shows the process of controlling connection with the GPIO pins.

Figure 11. Controlling Connections with the GPIO Pins


Embedded Microprocessor
WiFly Module
GPIO4 = High by WiFly Module
GPIO5 = High by Microprocessor
GPIO6 = High by WiFly Module

This signal instructs the microprocessor that it is OK to open a TCP connection.

Event: The WiFly module is associated with an access point and has a valid IP address.

Event: the microprocessor wants to open a TCP connection to the stored remote host.

The WiFly module is instructed to open a TCP/IP connection to the stored remote host.

This signal instructs the microprocessor that the TCP connection is open andit can begin sending/receiving data over the TCP link. To close the conenction, the microprocessor drives GPIO5 low.

Event: The WiFly module successfully opens the TCP connection to the stored remote host.

SETTING DEBUG PRINT LEVELS

You can enable print functions to assist with debugging the operation and status of the module. The **set sys printlvl** <*value*> command controls these additional print functions, where <*value*> is a bit-mapped register that controls which printout messages are sent to the UART. See “set sys printlvl <*value*>” on page 22 for more information.

Scan Output Format

You enable the scan output using the **set sys printlvl 0x4000** command. The scan output format differs, depending on the firmware you are running.

Firmware Version 2.36 & 2.45

Firmware version 2.36 and 2.45 support a comma-delimited scan output format, which a microprocessor can use to parse the RSSI information. The scan command output format is:

Index	Channel	RSSI	Security Mode	Capabilities	WPA Configuration	WPS Mode	MAC address	SSID

Where:

Field	Value
Index	2 character, decimal
Channel	2 character, decimal
RSSI	2 character, decimal (negative number)
Security mode	2 bytes (see Table 24)
Capabilities	Bit-mapped 4 hex bytes (see Table 25)
WPA Configuration	Bit-mapped 2 hex bytes (see Table 26)
WPS Mode	Bit-mapped 2 hex bytes (see Table 27)
MAC address	Address
SSID	Up to 32 chars

NOTE: The string **END** is added at the end of the scan data.

The following example shows the output format:

```
SCAN:Found 3
01,01,-59,04,1104,28,c0,20:4e:7f:08:df:85,dad-rules
02,03,-64,02,1104,28,00,00:30:bd:9b:49:22,basement
03,10,-71,04,3100,28,00,90:27:e4:5d:fc:a7,URSOMONEY
END:
```

Table 24 shows the security modes.

Table 24. Security Modes

Number	Description
0	OPEN
1	WEP (64 or 128)
2	WPA1
3	MIXED
4	WPA2
5	Enterprise WEP
6	Enterprise WPA1
7	Enterprise WPA mixed
8	Enterprise WPA2
9	Enterprise NO security

Table 25 describes the capabilities bit mask values.

Table 25. Capabilities Bit Mask Values

Bit Mask Value	Description
0004	Short slot time
0100	ESS (infrastructure mode)
0200	IBSS (ad hoc mode)
1000	Privacy (secure with WEP or WPA)
2000	Short preamble

Table 26 describes the WPA bit mask values.

Table 26. WPA Bit Mask Values

Bit Mask Value	Description
04	WPA_UNICAST_TKIP
08	WPA_UNICAST_AES_CCMP
10	WPA_BROADCAST_TKIP
20	WPA_BROADCAST_AES_CCMP

Table 27 describes the WPS bit mask values.

Table 27. WPS Bit Mask Values

Bit Mask Value	Description
02	WPS_PushButton_ACTIVE
40	WPS_SUPPORTED
80	WPS_PushButton_SUPPORTED

Firmware Version 2.22 through 2.30

Firmware version 2.22 through 2.30 supports a comma-delimited scan output format, which a microprocessor can use to parse the RSSI information. The scan command output format is:

Row Count	Channel	RSSI Value (dBm)	Security Mode	Capabilities	Access Point MAC Address	SSID

Example output from the scan command output is shown below:

```
SCAN:Found 5
01,01,-53,00,0200,1a:fc:90:e5:a5:37,QTDFW
02,01,-59,04,3104,00:15:f9:38:bd:b0,SensorNet
03,11,-72,04,3104,00:16:b6:45:63:98,CoolBox
04,11,-50,02,3100,00:18:02:70:7e:e8,airlink-11
05,11,-69,04,3100,00:14:6c:1f:f7:5e,ap-ssid-change-me
```

The security mode field for the this scan format is described below:

Security Mode	Open	WEP	WPA-PSK	WPA2-PSK	WPA-Enterprise	WPA2-Enterprise
Code	0	1	2	4	6	8

UART Heartbeat Messages

In firmware version 2.22 and higher, the module can output UART heartbeat messages. The bit-mapped message is output periodically while the module is in data mode and not connected to a remote host. Messages are not output while in command mode. The heartbeat message encodes the module's state for the embedded microprocessor. Based on the heartbeat message, the microprocessor can choose to change the configuration by going into command mode.

To enable the UART heartbeat messages, use the **set sys printlvl 0x10** command. The output of this mode is:

***8b30*8b30*8b30....**

Table 28 shows the output bit format.

Table 28. Output Bit Format

Bit	15..14	13..12	11..8	7..6	5	4	3..0
Function	Fixed	RESERVED	Channel	RESERVED	Authentication	Association	TCP status
Value	2 = Access point mode 3 = Ad hoc mode	Unused	0 - 13	Unused	1 = OK	1 = OK	0 = Idle 1 = Connected 3 = No IP 4 = Connecting 5 = Challenge for password

USING THE REAL-TIME CLOCK FUNCTION

The module's real-time clock keeps track of the number of seconds since the module was powered on and the actual time when the module synchronized with the sNTP time server. By default, the module keeps track of up time but does not synchronize with the time server because this synchronization requires the module to be associated with a network that can access the sNTP server. The real-time clock reads the time in seconds since 1970, which corresponds to the UNIX time.

In firmware version 2.23 and higher, you can set the RTC value in seconds using the **set time rtc** <*value*> command.

The default sNTP server is:

ADDR=129.6.15.28:123
ZONE=7 (GMT -7)

Use the **show time** command to see the current time and uptime as shown below:

```
<2.23> show t
Time=08:43:10
UpTime=10 s
```

To set the time, use the **time** command:

```
<2. 23> show t
Time NOT SET
UpTime=8 s
<2. 23> time
<2. 23> show t
Time=08:51:31
UpTime=15 s
```

NOTE: The module must be associated with a network for the module to contact the sNTP server.

Tthe module can also be configured to get the time whenever it powers up using the **set time enable 1** command. If you set the time enable to a value greater than 1, the module pulls the time continuously every <*value*> minutes.

For example, to configure the module to get time upon power up, see the following example:

```
<2. 23> set time enable 1
AOK
<2. 23> get time
ENA=1
ADDR=129.6.15.28:123
ZONE=7
```

To view a complete listing of the time variable, use the following command:

```
<2. 23> show t t
Time=09:02:10
UpTime=653 s
RTC=1293567548
Restarts=1
Wake=6
RAW=2345ab
```

NOTE: The RAW value is the 64-bit hex RAW value of the RTC, which ticks at 32,768 Hz.

TIME STAMPING PACKETS

You can use the time stamping feature to append 8 bytes to a TCP or UDP packet automatically. The **set ip flags 0x87** command enables the time stamp and keeps other default settings). The time stamp bits from MSB to LSB are:

User's TCP or UDP packet data	63..56	55..48	47..40	39..32	31..24	23..16	15..8	7..0

The 8 bytes represents the 64-bit raw value of the real-time clock register. The data is appended before calculating the TCP checksum so that the data passes through the TCP stack correctly. This register counts at 32,768 Hz. If the timeserver function is enabled, the RTC should accurately reflect the real time. This register also counts when the module is in sleep mode.

ADVANCED APPLICATIONS

This section describes a variety of advanced applications for the WiFly module, such as sending data using UDP, associating with access points, using the HTML client feature, upgrading the firmware over FTP, etc.

SENDING DATA USING UDP

UDP is a connectionless protocol: there is no initial handshaking between the hosts to set up the UDP connection and the receiver does not send an acknowledgement when it receives UDP packets. Therefore, UDP is an unreliable protocol because there is no guarantee that the data will be delivered correctly. However, because it is connectionless, UDP is suited for applications that cannot tolerate too much latency but can tolerate some errors in the data, e.g., video transmission.

To use UDP with the module, you must enable the UDP protocol using the **set ip proto 1** command. You must also specify the remote host's IP address and the local and remote port number that you will use for UDP communications. The following example shows the commands you use to enable UDP data transfer.

Example: Associate with a Network

set wlan ssid <*string*>	// Set the network name
set wlan phrase <*string*>	// Set the passphrase for WPA and WPA2 modes

Example: Set Up the Protocol & Port Number

set ip proto 1	// Enable UDP as the protocol
set ip host <*address*>	// Set the remote host's IP address
set ip remote <*value*>	// Set the remote port on which the host listens
set ip local <*value*>	// Set the port number on which the module listens
save	// Save the settings in the configuration file
reboot	// Reboot the module

NOTE: If you attempt to send data by typing characters on the keyboard or if your microcontroller is not sending data fast enough, the module sends out packets with fewer data bytes. To avoid this issue, set the flush timer to a higher value. By default, it is set to 10 ms. You can disable forwarding based on the flush timer (**set comm time 0**) or set it to a higher value (e.g., **set comm time 2000**).

Because UDP is a connectionless protocol, data begins flowing as soon as you reboot the module. Unlike TCP, you do not need to send an open command to establish the connection. The module acts like a data pipe: the UART data is sent over the Wi-Fi link via the UDP protocol (in this case) and the data coming from the Wi-Fi link (via UDP protocol in this case) is sent to the UART.

UDP Auto Pairing

With the UDP auto-pairing feature, the module temporarily stores the host IP address of the first remote device that sends a UDP packet to the module. This host IP address is stored in the module's RAM, which is cleared when the module sleeps or power cycles. This feature allows the module to echo to any client that sends a UDP packet.

Example: Turn on Auto Pairing

```
set ip host 0.0.0.0                                  // Set the host to 0.0.0.0
set ip flags 0x40
```

UDP Retry

This feature adds a level of reliability to the UDP protocol without adding the complete overhead of TCP protocol. When enabled, the module waits for a response on every UDP packet that is sent (any UDP packet coming back in). If the module does not receive the response packet by approximately 250 ms, the same UDP packet is sent out. This process continues until either:

- A UDP response is seen
- A new UDP packet is sent from the module and is acknowledged

Refer to "set ip flags <*mask*>" on page 16 for which bit to set to enable this feature.

UDP Broadcast

You can set up the module to generate UDP broadcast packets automatically, which is useful for the following reasons:

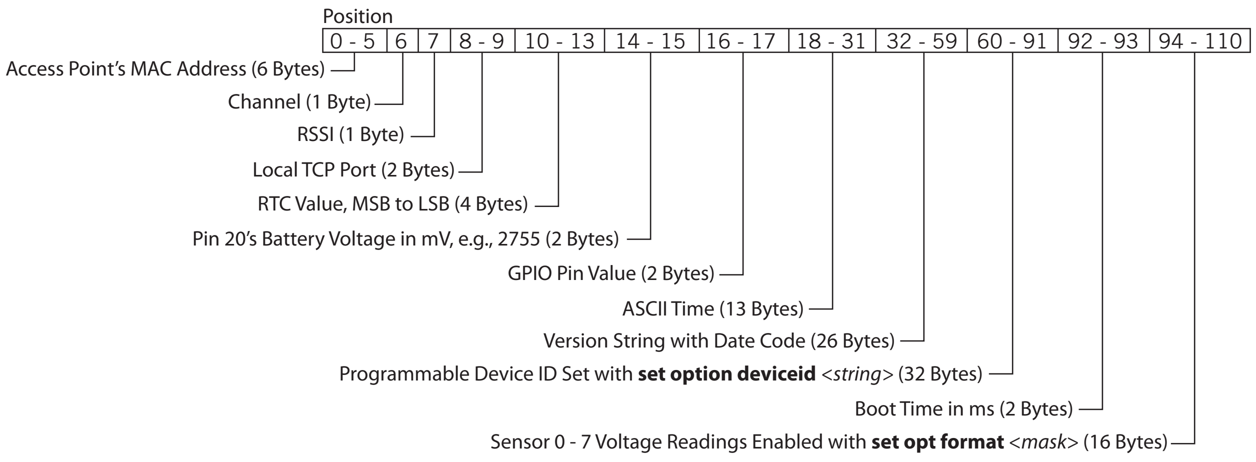
- Some access points disconnect devices that are idle. UDP broadcast informs the access point that the module is alive and wants to stay associated.
- Applications can use this feature to automatically discover and configure the module. If an application is listening for the UDP broadcast, a number of useful parameters are present in the package that can be used for auto-discovery. For example, the module's IP address and port number are part of the packet, thus an application can connect to the module and remotely configure it.
- The associated access point's MAC address, channel, and RSSI value are also available in this packet, enabling a simple location and tracking function.

By default, the module sends out a UDP broadcast to 255.255.255.255 on port 55555 at a programmable interval. You set the broadcast address, port, and interval using the **set broadcast** commands.

NOTE: You can send the module's sensor data out via UDP broadcast. The analog-to-digital convertor is 14 bits on a 400 mV signal, which translates to about 24 microvolts (0x61A80 in hex). When you use the **show q** command in command mode, the module displays the raw readings. However, for HTTP web posting and UDP broadcast packets, the module shifts the reading by 4 bits (which is a divide by 16) resulting in a 16-bit number. Therefore, if you want the actual voltage sampled, you must take the 16-bit number and shift it left by 4 bits to get the number of microvolts. If you the value in millivolts (and do not need high accuracy), right shift by another 6 bits, which is the same as dividing by about 1K.

The packet is 110 bytes of data as shown in Figure 12.

Figure 12. UDP Broadcast Packet Byte Format


Position
0 - 5
6
7
8 - 9
10 - 13
14 - 15
16 - 17
18 - 31
32 - 59
60 - 91
92 - 93
94 - 110
Access Point's MAC Address (6 Bytes)
Channel (1 Byte)
RSSI (1 Byte)
Local TCP Port (2 Bytes)
RTC Value, MSB to LSB (4 Bytes)
Pin 20's Battery Voltage in mV, e.g., 2755 (2 Bytes)
GPIO Pin Value (2 Bytes)
ASCII Time (13 Bytes)
Version String with Date Code (26 Bytes)
Programmable Device ID Set with **set option deviceid** <*string*> (32 Bytes)
Boot Time in ms (2 Bytes)
Sensor 0 - 7 Voltage Readings Enabled with **set opt format** <*mask*> (16 Bytes)

NOTE: To add sensor data to the UDP broadcast message you must enable the sensors using the sensor mask. The **set q sensor 0xff** command enables all sensors.

JOINING NETWORKS & MAKING CONNECTIONS

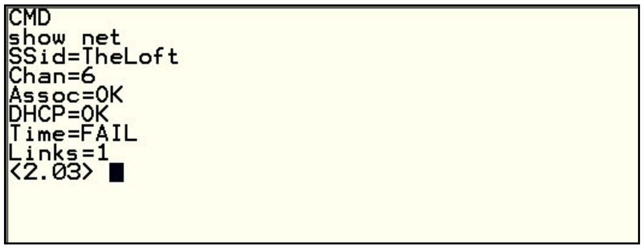
Configuring the module to make connections involves associating with an access point and opening a connection. Before you can configure the module over the WiFi link you must associate the module with a network and program the network settings. Therefore, the best method is to configure the module using the UART or over the air using ad hoc mode. This section describes how to configure the module over the UART using the RS-232 connector or an evaluation board. For this mode, open a terminal emulator on the COM port associated with the module. The default baud rate is 9,600, 8 bits, and no parity.

To configure using ad hoc mode, refer to "Ad hoc Networking Mode" on page 77. Once in ad hoc mode, open a telnet window using the IP address 169.254.1.1 port 2000.

Associate with an Access Point

From within the terminal window, put the module into command mode by typing **$$$**. The module responds with **CMD**, indication that it is in command mode. Type **show net** to display the current network settings as shown in Figure 13.

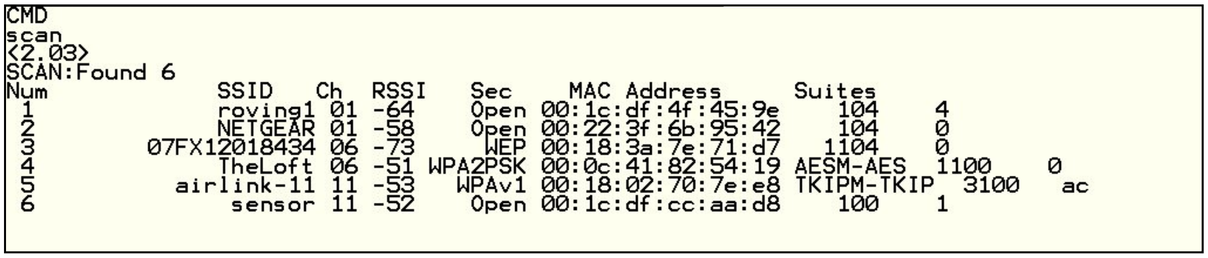
Figure 13. Display Current Network Settings



```
CMD
show net
SSid=TheLoft
Chan=6
Assoc=OK
DHCP=OK
Time=FAIL
Links=1
<2.03> █
```

Find all available networks with the scan command as shown in Figure 14.

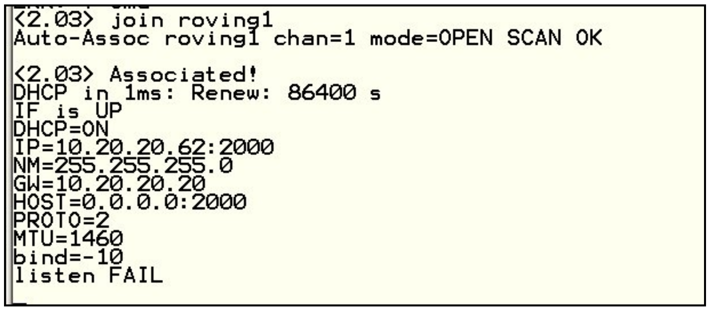
Figure 14. Find Available Networks



```
CMD
scan
<2.03>
SCAN:Found 6
Num              SSID   Ch  RSSI    Sec     MAC Address       Suites
 1            roving1  01  -64     Open  00:1c:df:4f:45:9e     104     4
 2            NETGEAR  01  -58     Open  00:22:3f:6b:95:42     104     0
 3       07FX12018434  06  -73      WEP  00:18:3a:7e:71:d7    1104     0
 4            TheLoft  06  -51  WPA2PSK  00:0c:41:82:54:19 AESM-AES   1100     0
 5         airlink-11  11  -53    WPAv1  00:18:02:70:7e:e8 TKIPM-TKIP  3100    ac
 6             sensor  11  -52     Open  00:1c:df:cc:aa:d8     100     1
```

If you are connecting to an open network, use the **join** command to associate with the access point. The scan list in Figure 14 shows that **roving1** is an open access point. Type **join roving1** (or **join # 1**) to associate with the network as shown in Figure 15.

Figure 15. Join the Network



```
<2.03> join roving1
Auto-Assoc roving1 chan=1 mode=OPEN SCAN OK

<2.03> Associated!
DHCP in 1ms: Renew: 86400 s
IF is UP
DHCP=ON
IP=10.20.20.62:2000
NM=255.255.255.0
GW=10.20.20.20
HOST=0.0.0.0:2000
PROTO=2
MTU=1460
bind=-10
listen FAIL
```

If the access point is secure, you must set the pass phrase prior to issuing the **join** command. The module attempts to inquire and determine the access point's security protocol; you do not need to set the authentication mode. To set the WPA pass phrase use the **set wlan phrase** <*string*> command. For WEP, set the key using the **set wlan key** <*value*> command.

Once the module has joined the network successfully, it stores the access point's SSID. You can save the SSID and the pass phrase to the configuration file so that the module can associate with the access point each time it boots.

Making Connections

To connect to the module, open an IP socket and connect to the module's IP address. You can use telnet to test the connection; type **open** <*address*> <*port*> in a telnet window. After the connection is open, you can type characters into the UART window and see them on the telnet window or vise versa.

Example: Open a Connection

```
open 10.20.20.62 2000                    // Open host shown in Figure 15
```

To make a connection from the module you need your server application's IP address and port number. A COM port redirector is a simple program you can use to test this functionality. This software opens an IP port and transfers all data it receives to a specified COM port on your machine. A free COM port redirector program for Windows is available from Pira at http://www.pira.cz/eng/piracom.htm.

In your COM port redirector program, note your computer's IP address, e.g., by typing the **ipconfig** command in the Microsoft Command Window. Go to your terminal emulator and put the module into command mode. Type the **open** <*address*> <*port*> command. The server reports that the connection is open and you can type characters into the UART window and see them on the server window or vice versa.

Setting Up Automatic Connections

Some applications require the module to connect to a remote server, send data, and then disconnect automatically upon power up (or wakeup). You can configure the module to perform this functionality automatically.

Set the network SSID and security, and set **autojoin** to 1. When the module wakes up or is powered on, the auto-connect timer causes the module to attempt a connection to the stored remote IP address and port. The sleep timer does not decrement while this connection is open and the idle timer does not decrement while data is flowing. When data stops for 5 seconds the connection is closed; the sleep timer puts the module in deep sleep. The wake timer begins the cycle again one minute later.

NOTE: You can also use ad hoc mode (**autojoin** 4); however, there will be a delay connecting to the ad hoc network from the remote computer. Therefore, make the sleep timer large enough to allow the network to get set up and the auto-connect to establish a TCP connection.

Example: Automatic Connection

```
set ip host  <address>         // Set up the remote machine's IP address
set ip remote_port <value>     // Set up the remote machine's IP port
set sys autoconn 1             // Automatically connect when ready
set com idle 5                 // Disconnect after 5 seconds with no data activity
set sys sleep 2                // Sleep 2 seconds after connection is closed
set sys wake 60                // Wake up after 1 minute of sleep
set uart mode 2                // Use UART data trigger mode, which causes the
                               // module to make a TCP/HTTP connection upon
                               // incoming UART data (supported in firmware version
                               // 2.19 and higher)
```

Controlling Connections using GPIO5 & GPIO6

You can use GPIO5 to control the TCP connection. After you configure the pin with the **set sys iofunc** command, the module attempts to connect to the stored IP address and port when GPIO5 goes high and disconnects when GPIO5 goes low.

Similarly, you can monitor the connection status by reading GPIO6. When it goes high, the connection is open; when it goes low, the connection is closed. Use the command **set sys iofunc** command to enable GPIO6.

Example: Use GPIO6 & GPIO6 to Control Connections

```
set sys iofunc 0x20            // Enable GPIO5
set sys iofunc 0x40            // Enable GPIO6
```

Using DNS Settings

The module contains a built-in DNS client. If you do not specify the host's IP address, (i.e., it is set to 0.0.0.0), the module uses DNS protocol. When you set the host name using the **set dns name** <*string*> command, the module automatically attempts to resolve the host address. When the address is resolved, the module connects automatically.

To manually look up a host's IP address, use the **lookup** <*string*> command, where <*string*> is the hostname.

Example: Use DNS

set dns name my_server	// Set the DNS host name to my_server

Using the Backup IP Address/Connect Function

The module contains a feature for auto-retry and redundancy. If the host's first IP address connection fails, the module uses the backup IP (if set). If this fails (or is not set), the module uses the first DNS name. If this fails (or is not set), the module uses the backup DNS name (if set).

Example: Set the Backup IP Address

set ip backup <*address*>	// Set the backup IP address

Example: Set the Backup DNS Name

set dns backup <*string*>	// Set the backup host name

USING THE HTML CLIENT FEATURE

The module has a built-in HTML client. When enabled, the module can get or post data to a web server. For example, you can use the HTML client to post serial and/or sensor data to the host web server. This feature makes possible to provide Wi-Fi capabilities to applications such as GPS units, remote sensors, weather stations, etc.

Example: Retrieve Web Server Data

In this example, you want to retrieve data from the web server with the format:

http://www.webserver.com/ob.php?obvar=WEATHER

To perform this function, use the following settings:

set ip proto 18	// Enable the HTML client
set dns name www.webserver.com	// Set the web server name
set ip address 0	// Turn on DNS
set ip remote 80	// Set the web server port, 80 is standard
set com remote 0	// Turn off the REMOTE string so that it does not // interfere with the post

To make the connection, use the **open** command or you can use **open www.webserver.com 80**. The user's microprocessor writes the following string to the UART:

GET /ob.php?obvar=WEATHER \n\n

Where the **\n** is the linefeed character (decimal 10 or hex 0xa). Two linefeeds are required for the web server to know the page is complete.

NOTE: Some web servers require a carriage return and linefeed to indication the page is complete. In this case, use **\r\n** at the end of the string instead of **\n\n**.

Built-In HTML Client Modes

You can set up the module to post data to and get data from a web server automatically without an external host CPU. You enable these advanced web features using the **set option format** <*flag*> command, where <*flag*> represents a bit-mapped register. Refer to the "set opt format <*flag*>" command on page 18 for the bit function descriptions. Table 29 describes the wake reason values.

Table 29. Wake Reason Values

Value	Wake Reason
0	Undefined.
1	Power on or hardware reset (battery install or power up).
2	Sleep (wake when the sleep timer is expired).
3	Sensor.
4	Undefined.
5	Button (RN-370 serial adapter only).
6	Software reboot.
7	Watchdog.

Example: HTML Client Modes

```
set option format 1          // Automatically send an HTML data header
set option format 7          // Append sensor data in ASCII hex format
set option format 11         // Append all key value pairs to the sensor data
```

Connect to a Web Server Automatically

You can configure the module to post data to a webserver automatically using the **set sys auto** <*value*> command, where <*value*> is a decimal number representing seconds. For example, you can configure the module to connect to the web server every 10 seconds with the **set sys auto 10** command.

When HTTP mode is set, the module automatically appends two linefeeds (**\n\n**) to the end of the packet.

NOTE: If the HTML header contains spaces, you must use the $ character to indicate spaces in the string. (A space is the command delimiter.) When the module's command parser sees the $, it converts it to a space character.

Example: Connect to Web Server Every 30 Seconds

Use the following commands to configure the module to connect to a web server every 30 seconds:

```
set com remote  GET$/ob.php?obvar=WEATHER        // Setup the HTML string
set sys auto 30                                  // Auto-connect every 30 seconds.
set option format  1                             // Send header automatically when connection is open
set ip proto  18                                 // Turn on HTTP mode = 0x10  + TCP mode = 0x2
```

Connect to a Web Server Automatically when UART Data Is Received

The module supports a mode in which it can connect to the web server when it receives UART data.

NOTE: If you attempt to send data by typing characters on the keyboard or if your microcontroller is not sending data fast enough, the module sends out small packets of data (it sends out many packets of small MTU size). To avoid this issue, set the flush timer to a higher value, e.g., **set comm time 5000**. By default, it is set to 10 ms.

Example: Connect to Web Server when UART Data Is Received

```
set ip proto 18                                  // Turn on HTTP mode = 0x10 and TCP mode = 0x2
set dns name www.webserver.com                   // Set the web server name
set ip host 0                                    // Turn on DNS
set ip remote 80                                 // Set the web server port, 80 is standard
set com remote GET$/userprog.php?DATA=           // Sample server application
set uart mode 2                                  // Automatically connect using data trigger mode
```

When the serial UART data comes in, the module automatically connects to the web server, and sends:

GET /userprog.php?DATA= <*users serial data*> **\n\n**

Post Binary Data

Web servers expect ASCII data. If the user data is binary, the module can convert it to ASCII format before sending it to the web server.

Example: Convert Data from Binary to ASCII

```
set ip proto 18                                  // Turn on HTTP mode = 0x10  and TCP mode = 0x2
set dns name www.webserver.com                   // Set the web server name
set ip host 0                                    // Turn on DNS
set ip remote 80                                 // Set the web server port, 80 is standard
set com remote GET$/userprog.php?DATA=           // Sample server application
set option format 1                              // Convert binary data to ASCII hex format
```

For example, if the incoming UART data is 6 bytes of binary data with hex values 0x01, 0xAB, 0x03, 0xFF, 0x05, and 0x06, the module sends this string **GET /userprog.php?DATA=01AB03FF0506\n\n** to the web server.

Post Sensor Data Automatically

The module can send the value of the GPIO and sensor pins to the web server automatically. The data arrives as 18 bytes of ASCII hex data in the format *<2 bytes GPIO><channel 0 thru 7 sensor data>*.

NOTE: The analog-to-digital convertor is 14 bits on a 400 mV signal, which translates to about 24 microvolts (0x61A80 in hex). When you use the **show q** command in command mode, the module displays the raw readings. However, for HTTP web posting and UDP broadcast packets, the module shifts the reading by 4 bits (which is a divide by 16) resulting in a 16-bit number. Therefore, if you want the actual voltage sampled, you must take the 16-bit number and shift it left by 4 bits to get the number of microvolts. If you the value in millivolts (and do not need high accuracy), right shift by another 6 bits, which is the same as dividing by about 1K.

Example: Post Sensor Data to Web Server

```
set ip proto 18                             // Turn on HTTP mode = 0x10  and TCP mode = 0x2
set dns name www.webserver.com              // Set the web server name
set ip host 0                               // Turn on DNS
set ip remote 80                            // Set the web server port, 80 is standard
set com remote  GET$/userprog.php?DATA=     // Sample server application
set q sensor 0xff                           // The module samples all 8 sensor channels
set sys auto 30                             // Connect every 30 seconds
set option format  7                        // Send the header plus the sampled binary data
                                            // converted to ASCII format
```

The resulting string sent to the server is:

GET /userprog.php?DATA=0F30000011112222333344445555666677777\n\n

The data format for this example is:

2 Bytes GPIO	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
0F30	0000	1111	2222	3333	4444	5555	6666	7777

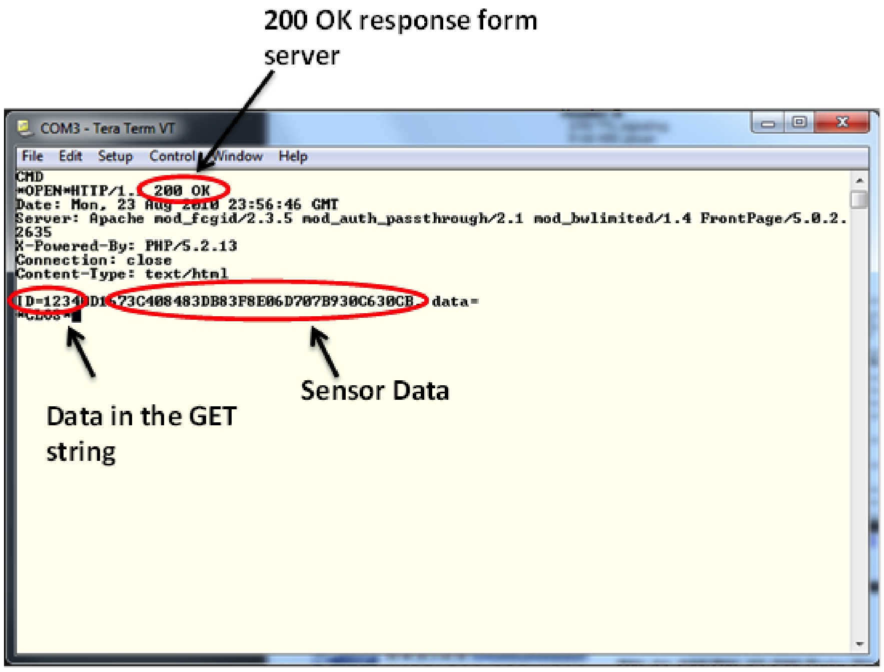
HTML Client Example: Posting Sensor Data Automatically

In this example, the module connects to the web server at www.rovingnetworks.com/server.php?value= and posts the sensor data to the web server every 60 seconds. You set the network connections as described previously, and set additional parameters.

```
set ip proto 18                             // Turn on HTTP mode = 0x10  and TCP mode = 0x2set
dns name www.rovingnetworks.com             // Set the web server name
set ip host 0                               // Turn on DNS
set ip remote 80                            // Set the web server port, 80 is standard
set com remote GET$/server3.php?value=      // Set up the server application string
set sys auto 10                             // Automatically connect every 10 seconds
set option format 7                         // Send the header and sampled binary data converted
                                            // to ASCII
set q sensor 0xFF                           // Set the sensor mask to sample all channels
save                                        // Save the configuration to the config file
reboot                                      // Reboot so that the settings take effect
```

After issuing these commands, the web server returns a 200 OK message, as shown in Figure 16.

Figure 16. Server Response


200 OK response form server
COM3 - Tera Term VT
File Edit Setup Control Window Help
Data in the GET string
Sensor Data

```
CMD
*OPEN*HTTP/1.  200 OK
Date: Mon, 23 Aug 2010 23:56:46 GMT
Server: Apache mod_fcgid/2.3.5 mod_auth_passthrough/2.1 mod_bwlimited/1.4 FrontPage/5.0.2.
2635
X-Powered-By: PHP/5.2.13
Connection: close
Content-Type: text/html

ID=1234 D1 73C408483DB83F8E06D707B930C630CB  data=
*CLOS*
```

You can view the data you sent to the Roving Networks web server at http://rovingnetworks.com/wiflys/view.

HTML Client Example: Posting UART Data to a Web server

The module can post serial UART data in ASCII or binary format automatically. In this example, when the serial UART data comes in, the module connects and sends data to the web server in the following format:

GET /server.php?value=<user serial data> \n\n

Use the following commands to set the parameters:

set ip proto 18	// Turn on HTTP mode = 0x10 and TCP mode = 0x2set
dns name www.rovingnetworks.com	// Set the web server name
set ip host 0	// Turn on DNS
set ip remote 80	// Set the web server port, 80 is standard
set com remote GET$/server3.php?value=	// Set up the server application string
set sys auto 10	// Automatically connect every 10 seconds
set option format 1	// Send a HTML header
set uart mode 2	// Connect automatically using data trigger mode
save	// Save the configuration to the configuration file
reboot	// Reboot so that the settings take effect

With these settings enabled, the module connects to the web server every time it receives data on the RX line. Serial data is sent to the host web server according to the flush timer and the flush size.

NOTE: You cannot append the sampled sensor data to the UART data. Enabling **option format 7** with **set uart mode 2** results in erroneous data.

You can view the data you sent to the Roving Networks web server at http://rovingnetworks.com/wiflys/view.

UPGRADING FIRMWARE VIA FTP

The module has a file system for storing firmware and configuration files. Use the **ls** command to view files. The file size is displayed in sectors and the active boot image is identified in the final message. For example:

```
FL#     SIZ     FLAGS
 11     18      3               WiFly_GSX-2.21
 29     1       10              config
190 Free, Boot=11, Backup=0
```

You can store multiple firmware images and configuration files.

NOTE: The module's flash file system only is used to store firmware and configuration files. Currently, the file system cannot be used to store data files.

The module contains a built-in FTP client for downloading files and updating the firmware. The client uses passive mode FTP, which allows operation through firewalls and the Internet. To connect to Roving Networks to obtain the latest released firmware, use the following settings:

FTP server: **rn.microchip.com**
FTP username: **roving**
FTP password: **Pass123**
FTP filename: **wifly_GSX-**<*version*>**.img** (RN-131) or **wifly_EZX-**<*version*>**.img** (RN-171)
FTP directory: **./public** (this parameter cannot be modified)

NOTE: Before using FTP to upgrade the firmware, the module must first be associated with an access point that is connected to the Internet.

To update the firmware, issue the command **ftp update** <*string*>, where <*string*> is an optional filename (use the optional name to bypass the default firmware file name)

The module retrieves the file and switches the boot image to the new file, resulting in the following messages:

```
<2.20> ftp update
<2.20> FTP connecting to 208.109.78.34
FTP file=30
.....................................................................
FTP OK.
```

NOTE: After the module reboots with the new firmware, Roving Networks recommends that you reset the module to the factory default parameters using the **factory R** command. Failure to do so may result in some variables being initialized with random values.

The previous firmware becomes the backup image. The following example shows the file system after a successful update:

```
FL#    SIZ    FLAGS
 11    18     3           WiFly_GSX-2.20
 29    1      10          config
 30    18     3           WiFly_GSX-2.21
208 Free, Boot=30, Backup=11
```

After downloading, the firmware checks the image and compares it to the stored values in the file before committing the image to flash and updating the boot record. If the checksum fails, the module displays **UPDATE FAILED=x** and deletes the image.

NOTE: You must reboot or power cycle the module to use the new firmware. To boot with different firmware, use the command **boot image** <*value*>, which sets the current boot image as <*value*>.

For example, to boot the previous image using the previous example:

```
<2.20> boot image 11
Set Boot Image 11, =OK
```

NOTE: After changing the boot pointer to the new image, you must reboot the module to boot up with the new image. Once the module boots up with the new image, perform a factory reset on the module to initialize all the parameters to the factory default settings. Then, you can reinitialize the parameters as required.

FTP CLIENT

In addition to downloading firmware via FTP, with firmware version 2.22 and higher, the module can get and put files to an FTP server.

Connect to an FTP Server

By default, the module is configured to download the latest firmware from the Roving Networks' FTP server. To configure the module to connect to your own FTP server, you must adjust the parameters as described in the example below.

Example: Connect to an FTP Server

```
set ftp address <address>          // Set FTP server's IP address. Default is 208.109.78.34
set ftp dir <string>               // Set the directory in the FTP server. Default is public.
set ftp user <string>              // Set the user name
set ftp pass <string>              // Set the password
save                               // Save the settings
reboot                             // Reboot the module
```

NOTE: This example assumes that the FTP server is already set up and configured correctly and that the module is already configured to associate with a wireless network.

Creating Files on the FTP Server

Once the module is configured to connect to the FTP server, it can create files on the FTP server. To create a file, you use the **ftp put** <*filename*> command, where <*filename*> is up to 64 bytes. This command creates a file on the FTP server with the name <*filename*> and prints the open string on the UART. By default, the open string is ***OPEN***. After you see ***OPEN*** on the UART, you can begin writing data in to the file.

There are two options to close the file:

- Send the close string, which is ***CLOS*** by default.
- Use the FTP close timer with the command **set ftp timer** <*value*>. Once you finish writing to the file, this timer begins counting down and closes the file when the timer gets to zero. The timer is one eighth of <*value*>. For example, to set a 5-second timer, the command is **set ftp timer 40**.

The open and close stings are configurable using the following commands:

set comm open <*string*>	// Set the open string
set comm close <*string*>	// Set the close string

Example: Put File on FTP Server

ftp put demo.txt	// Upload the file demo.txt
set ftp timer 40	// Close the connection 5 seconds after file uploads

Retrieving Files from the FTP Server

The module can retrieve files from the FTP server. The retrieved file is not stored in module's flash memory; the module acts as a transporter and passes the file over the UART interface as the file is being transferred.

To retrieve a file from the FTP server issue the **ftp get** <*filename*> command. The module prints the open string on the UART and the file begins transferring from the FTP serer to the module. When the file transfer complete, the module prints the close string indicating the file is transferred and the FTP connection is closed.

Example: Retrieve File from FTP Server

ftp get demo.txt	// Download the file demo.txt from the FTP server

WI-FI PROTECTED SETUP (WPS)

Wi-Fi Protected Setup (WPS) is a standard for easy and secure establishment of a wireless home network. This standard was created by the Wi-Fi Alliance and officially launched on January 8, 2007.

The goal of the WPS protocol is to simplify the process of configuring security on wireless networks. The protocol is meant to allow home users who know little of wireless security and may be intimidated by the available security options to configure Wi-Fi Protected Access, which is supported by all newer Wi-Fi certified devices (but not older Wi-Fi devices).

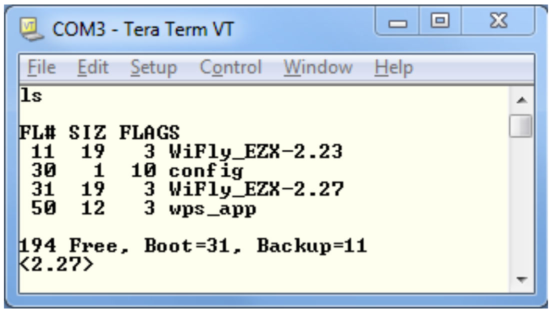
The most common mode of WPS is the Push Button Mode (PBC) in which the user simply pushes a button on both the access point and the wireless client (e.g., the Roving Networks' WiFly module). See Figure 17.

Figure 17. Push-Button WPS

The module supports the WPS feature in firmware version 2.28 and higher. To upgrade to the current firmware version and download the WPS application, refer to the WPS application note on the Support page of the Roving Networks web site at http://www.rovingnetworks.com/Support_Overview.

NOTE: Modules that ship with firmware version 2.28 or higher already have the WPS application. You can confirm whether your module has the application using the **ls** command. See Figure 18.

Figure 18. Confirming WPS Application is Installed



```
ls

FL# SIZ FLAGS
 11  19    3 WiFly_EZX-2.23
 30   1   10 config
 31  19    3 WiFly_EZX-2.27
 50  12    3 wps_app

194 Free, Boot=31, Backup=11
<2.27>
```

Launching a WPS Application

There are two ways to invoke a WPS function:

- Using the **wps** command in the console.
- Using the ad hoc/factory reset pin (GPIO9).

To invoke a WPS function using the ad hoc/factory reset mode:

1. Enable the WPS function on GPIO9 using the **set system trigger 0x10** command. WPS on GPIO9 is disabled by default to avoid accidentally invoking the WPS function.
2. The WPS application is invoked when GPIO9 goes from low to high. You can enable this mode on the RN-134 and RN-174 boards by installing and removing the ad hoc/factory reset jumper.

When the WPS application exists, it reboots the module to associate with the WPS-enabled access point. If GPIO9 is high, the module boots in ad hoc mode. Care must be taken to drive GPIO9 low before the module reboots. A good indicator is the red LED on the RN-134 and RN-174 boards. When this LED flashes, indicating the module is scanning for a WPS-enabled access point, you should drive GPIO9 low.

By default, the WPS code prints messages on the UART as it scans channels, detects access points, and tries to complete WPS. You can disable these messages using the **set sys print 0** command.

Status LEDs during WPS Process

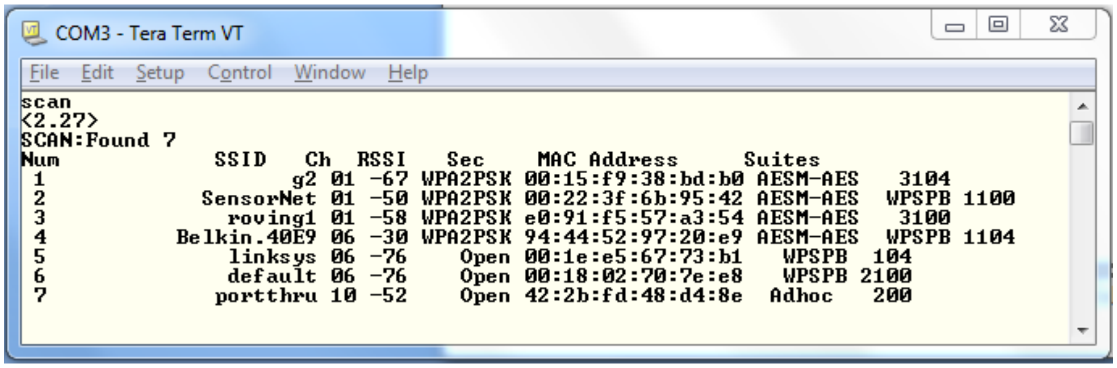
In WPS mode, the LEDs indicate activity:

- The red LED flashes while the module is scanning for WPS-enabled access points.
- The yellow LED goes on solid while negotiation is in progress with a WPS-enabled access point. If the process is successful, the WPS application quits and the module reboots.
- If the module is set to use the standard GPIO functions (i.e., not the alternate GPIO4 functions), the green LED blinks once per second. If the alternate GPIO4 function is enabled, the green LED goes high.

Scan Output Format Showing WPS-Enabled Access Point

The scan output shows access points that support the WPS feature. As shown in Figure 19, access points that support WPS are listed with WPSPB in the security suites.

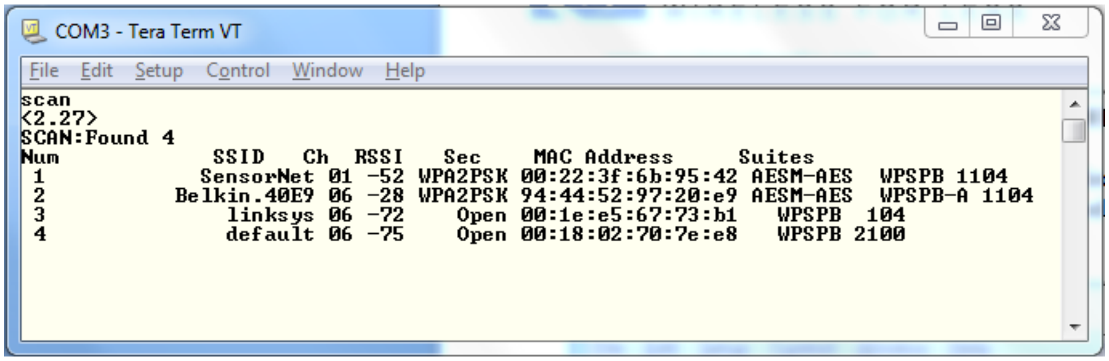
Figure 19. Access Points Supporting WPS in Scan List



```
scan
<2.27>
SCAN:Found 7
Num              SSID   Ch  RSSI   Sec     MAC Address      Suites
 1                 g2  01 -67 WPA2PSK 00:15:f9:38:bd:b0 AESM-AES   3104
 2          SensorNet  01 -50 WPA2PSK 00:22:3f:6b:95:42 AESM-AES  WPSPB 1100
 3            roving1  01 -58 WPA2PSK e0:91:f5:57:a3:54 AESM-AES   3100
 4         Belkin.40E9 06 -30 WPA2PSK 94:44:52:97:20:e9 AESM-AES  WPSPB 1104
 5            linksys  06 -76    Open 00:1e:e5:67:73:b1   WPSPB  104
 6            default  06 -76    Open 00:18:02:70:7e:e8   WPSPB 2100
 7           portthru  10 -52    Open 42:2b:fd:48:d4:8e  Adhoc    200
```

If you press the WPS button on the access point and then perform a scan, the scan returns a **–A** to indicate the access point is in WPS active mode. See Figure 20.

Figure 20. Scan Showing Access Points in WPS Active Mode



```
scan
<2.27>
SCAN:Found 4
Num              SSID   Ch  RSSI   Sec     MAC Address      Suites
 1          SensorNet  01 -52 WPA2PSK 00:22:3f:6b:95:42 AESM-AES  WPSPB 1104
 2         Belkin.40E9 06 -28 WPA2PSK 94:44:52:97:20:e9 AESM-AES  WPSPB-A 1104
 3            linksys  06 -72    Open 00:1e:e5:67:73:b1   WPSPB  104
 4            default  06 -75    Open 00:18:02:70:7e:e8   WPSPB 2100
```

AD HOC NETWORKING MODE

There are two types of networks, infrastructure and ad hoc. Infrastructure networks, in which an access point links all Wi-Fi devices, are the most common. The access point keeps track of devices on the local network and directs IP packets. In many cases, the access point is also a router and forwards packets from the local network to other networks and the Internet. It is also very common for the access point to run a DHCP server, which tracks and assigns IP addresses.

Ad hoc networks are point-to-point networks in that each Wi-Fi device is linked directly to every other Wi-Fi device on the ad hoc network. There is no access point. All Wi-Fi devices on the ad hoc network participate in keeping the network alive and each keeps track of the other active devices on the network by sending and receiving beacon and probe packets. In most cases, IP addresses are assigned through automatic IP, although one of the Wi-Fi devices can be configured as a DHCP server.

NOTE: Roving Networks supports ad hoc networking, however, going forward, ad hoc mode will be replaced with soft AP mode. Ad hoc mode and soft AP mode are mutually exclusive and cannot operate at the same time. The support for these modes resides in separate firmware images loaded on the module. By default, Roving Networks modules are shipped with the ad hoc mode image to maintain backwards compatibility with existing applications. Refer to "Access Point (AP) Mode" on page 42 for more information on AP mode.

Configuring Ad Hoc Mode

You can configure the module to setup an ad hoc network. This mode is useful for point-to-point communications. When in ad hoc mode the device appears like an access point with which other Wi-Fi devices can associate.

NOTE: Currently the module only supports the OPEN mode for creating ad hoc networks.

You can enable ad hoc mode via hardware or software commands.

Enable Ad Hoc Mode in Hardware

To enable ad hoc mode using hardware, set GPIO9 high (3.3 V) at power up. For the RN-134 board, GPIO9 is on pin 1 on the jumper block (J2). For the RN-174 board, GPIO9 is on the J6 connector. Upon power up with GPIO9 high, the WiFly module creates an ad hoc network with the following settings:

SSID:	WiFly-GSX-*XX*, where *XX* is the final two bytes of the device's MAC address
Channel:	1
DHCP:	OFF
IP address:	169.254.1.1
Netmask:	255.255.0.0

With the ad hoc jumper in place, these settings override any current saved configuration settings.

Enable Ad Hoc Mode in Software

To enable ad hoc mode in software, you use the **set wlan** command with the **join**, **ssid**, and **chan** parameters. For example, type the following commands in command mode:

```
set wlan join 4
set wlan ssid my_adhoc_network
set wlan chan 1
```

Turn off DHCP so that the module does not attempt to obtain an IP address from another device, and set the module's IP address and netmask. Because automatic IP assignment fixes the first two bytes of the IP address, use 255.255.0.0 as the netmask so that other devices connecting to the module can be reached. You can also set the netmask to a smaller subnet if the other device's IP addresses begin statically at the same subnet as the ad hoc device.

```
set ip address 169.254.1.1
set ip netmask 255.255.0.0
set ip dhcp 0
```

Save your configuration and reboot. The module will be in ad hoc mode.

The module can associate with an ad hoc network created by another device. Type the commands:

```
set wlan ssid my_adhoc_network
save
reboot
```

To associate with an ad hoc network without saving the changes to the module's flash memory, use the **join** command, e.g., **join my_adhoc_network** <cr>. (If the module was already associated with another network, you must first disassociate with it using the **leave** command.)

If DHCP is enabled, the WiFly device obtains an IP address automatically when it associates with the ad hoc network. By definition, auto IP sets the first two bytes of the subnet to 169.254.*xxx.xxx*. The WiFly device requires 2 to 3 seconds to resolve the IP address.

To set the IP address statically, disable DHCP and explicitly assign the IP address:

```
set ip dhcp 0
set ip address 169.254.1.2
```

You can confirm that the device has properly associated with the ad hoc network using the **ping** keyword:

```
ping 169.254.1.1 10
```

You can associate with the ad hoc network from a computer by specifying the network name (and password, if required) in the operating system. For example, choose Control Panel > Networking and Sharing > Networking and Sharing Center (Windows Vista) or Control Panel > Network Connections (Windows XP). You can then view available networks and select the name of the WiFly ad hoc network.

NOTE: Once associated with the ad hoc network, Windows Vista may require a few minutes to allocate an IP address. To work around this issue, assign a static IP address under Network Settings > TCP/IP > Properties.

Once your computer is associated with the ad hoc network, you can use the module's IP address to open a connection or connect using telnet as you would with an enterprise connection.

Scanning for Access Points in Ad Hoc Mode

The module supports ad hoc and infrastructure network modes, but not simultaneously. Scanning for wireless networks is a function of infrastructure mode. Therefore, the module disables ad hoc mode before scanning.

With firmware version 2.22 and higher, the module can scan for networks while in ad hoc mode. Issuing the scan command temporarily disables ad hoc model while the module is scanning. Ad hoc mode is restored automatically when the scan completes. If you are connected to the module over telnet, the scan result is sent over telnet and ad hoc mode is restored.

ANALOG SENSOR CAPABILITY

The module has 8 analog sensor inputs that can be driven between 0 to 1.2-V DC. You can sample the analog inputs and read digital value using the **show q** <*value*> command, where <*value*> is a decimal number representing the channel. See "show q <*value*>" on page 36 for more details.

WARNING: Driving these inputs above 1.2 V can permanently damage the module.

The channel is the analog sensor input from 0 to 7. The value for the analog sensor input is measured in microvolts and is returned as 8*xxxxx*, where the beginning 8 is a start marker.

You can also sample multiple channels by using a bit mask using the **show q 0x1**<*mask*> command, where <*mask*> is a bit mask of the channels. See "show q 0x1<*mask*>" on page 36 for more details.

Example: Read Channels 0, 1 & 7

```
show q 0x183                                 // Read channels 0, 1, and 7
```

The results are in the format **8**<channel 0>**, 8**<channel 1>**, 8**<channel 7>**\r\n**

The analog input hardware specification is:

Input voltage range:	0 - 1.2 V, however, the analog-to-digital converter saturates at 400 mV
Resolution:	14 bits = 12 uV
Sampling frequency:	35 us
Accuracy:	5% un-calibrated

The accuracy of each analog sensor reading can be offset by up to 5% due to variations from chip to chip. To improve accuracy, Roving Networks recommends using a precision reference voltage on one of the analog inputs to calculate the offset. The offset is the same for all analog inputs. For example:

- Drive precision 200 mV reference on analog input 4.
- Read analog input 4 and compute the offset.

If you read 210 mV you know that the offset is +10 mV. When you read input 5, subtract 10 mV from the result.

Sampling Sensor Pins Automatically

The sensor pins can be sampled automatically and data forwarded in two modes:

- The UDP broadcast packet can contain the sample values.
- In HTTP mode, the sampled pin data can be forwarded to a remote server

To enable these modes, use the **set q sensor** <*mask*> command.

Example: Sample All Sensor Inputs

```
set q sensor 0xff                            // Sample all sensor inputs
```

Using the Built-In Sensor Power

The modules contain an on-board sensor power pin, which is controlled by the **set q sensor** <*mask*> command. <*mask*> is a bit mask value that determines which sensor pins to sample when sending data using the UDP broadcast packet or the HTTP auto-sample function. See "set q sensor <*mask*>" on page 19 for more details.

NOTE: In versions of firmware prior to 2.23, this command is named **set option sensor**.

Firmware versions 2.23 and higher support the **set q power** <*value*> command. This command sets an 8-bit register with two 4 bit nibbles that automatically turns on the sensor power. See "set q power <*value*>" on page 20 for more details on using this command.

DEFAULT CONFIGURATION

This section describes the default configuration settings and how to restore them.

ADHOC PARAMETERS

Beacon	100 (milliseconds) for ad hoc mode only
Probe	5 (seconds to look for beacons before declaring ad hoc is lost) for ad hoc mode only
Reboot	0 (unused parameter for future development. Leave at default value.)

BROADCAST PARAMETERS

IP address	255.255.255.255
Port	55555
Interval	7 (seconds)

COMM PARAMETERS

Close string	*OPEN*
Open string	*CLOS*
Remote string	*HELLO*
Flush size	64
Match character	0
Flush timer	10 (milliseconds)
Idle timer	0
Cmd char	$

DNS PARAMETERS

IP address	0.0.0.0
Name	dns1
Backup	rn.microchip.com
Lease	86400 for ad hoc mode only

FTP PARAMETERS

Server address	0.0.0.0
File	wifly_GSX-<*version*>.img (RN-131), wifly_EZX<*version*>.img (RN-171)
User	roving
Password	Pass123
Dir	public

Timeout	200
FTP_mode	0x0

IP PARAMETERS

DHCP	ON (1 = enabled)
IP address	0.0.0.0
Net mask	255.255.255.0
Local port	2000
Gateway	0.0.0.0
Host	0.0.0.0
Remote port	2000
Protocol	2 (TCP server and client)
MTU	1524
Flags	0x7
TCP mode	0x0
Backup	0.0.0.0

OPTIONAL PARAMETERS

Device ID	WiFly-GSX
Join timer/WPA timer	1000
Replacement char	$ (0x24)
Format	0x00
Password	“” (no password enforced)
Signal	0
Average	5

SYSTEM PARAMETERS

Sleep timer	0
Wake timer	0
Trigger	0x1 (SENS0 pin wakes up the device)
Auto connect	0
IOfunc	0x0 (No alternate functions)
IOmask	0x20F0 (for RN-131) / 0x21F0 (for RN-171)
IOvalue	0x0
Print level	0x1 (Print enabled)
Debug Register	0x0 (Unused parameter for future development. Leave at default value)

TIME SERVER PARAMETERS

Enable	0 (disabled)
Server address	129.6.15.28 (fixed to port 123 - SNTP protocol)
Zone	7 (Pacific time, USA)

UART PARAMETERS

Baudrate	9600
Flow	0 (disabled)
Mode	0
Cmd_GPIO	0

WLAN PARAMETERS

SSID	roving1
Channel	0 (Automatic scan)
External antenna	0 (Off - use on-board chip antenna for RN-131 ONLY)
Join mode	1 (Automatically scan and join based on SSID) for firmware version 2.36 (ad hoc mode) and lower 0 for firmware version 2.45 (soft AP mode) and higher
Authentication mode	OPEN
Mask	0x1FFF (All channels)
Rate	12 (24 Mbit)
Linkmon	0
Passphrase	rubygirl
TX Power	0 (which implies 12 dBm. Applicable to RN-171 module only)

STRING VARIABLE SIZES

Table 30 provides the string variable sizes for the following parameters:

Table 30. String Variable Sizes

Parameter	Value (Bytes)
FTP Parameters	
file	32
user	16
pass	16
dir	32
wlan Parameters	
ssid	32
phrase	64
DNS Parameters	
DNS host name	64
DNA backup host name	64
comm Parameters	
open	32
close	32
remote	64
deviceid	32

RESTORING DEFAULT CONFIGURATION SETTINGS

You can restore the default factory configuration settings in software and hardware.

- *Software*—In command mode, use the factory RESET command to restore the defaults. This command automatically loads the default settings and executes a save command. Next, send the reboot command so that the module reboots with the default configuration.
- *Hardware*—Set GPIO9 high on power up to arm the factory reset function. Then toggle GPIO9 five (5) times, which restores the configuration to the factory reset. GPIO9 is sampled at about 1 Hz; therefore, if you are using a CPU to generate the signal, make sure that GPIO9 transitions (high to low or low to high) are at least 1 second long.

You can specify a user configuration file as the factory reset settings. Prior to this firmware version only the hardcoded factory defaults would be restored. If you have stored a configuration file named **user**, the module reads it as the factory default instead of using the factory hardcoded defaults. If no **user** configuration file is present, the module uses the hardcoded factory defaults.

You create the **user** configuration file using the **save user** command, which saves the current configuration settings into a file named **user**.

Even if a user configuration file exists, arming and toggling GPIO9 7 times overrides the user settings and restores the module to the factory hardcoded defaults. This bypass mechanism allows you to restore the factory defaults in case a bad configuration is saved into the user file.

Issuing the **factory RESET** command while in command mode restores the module to a factory default state.

NOTE: You must reboot the module or reset it for the new settings to take effect.

BOOT-UP TIMING VALUES

Table 31 shows the boot-up timing values.

Table 31. Boot-Up Timing Values

Function	Description	Time (ms)
Power up	Power up time from reset high or power good to boot code loaded.	70
Initialization	Initialize ECOS.	50
Ready	Load configuration and Initialize application	30
Join	Associate using channel = 0 (full channel scan, mask = 0x1FFF).	80
	Associate using channel = 0 (primary channel scan, mask = 0x421).	15
	Associate using channel = X (fixed channel).	5 - 20
Authentication	Authenticate using WPA1 or WPA2 (highly dependent on access point response).	50 - 250
Aquire IP	DHCP obtain IP address (highly dependent on DHCP server response time).	AP dependent

SUPPORTED ACCESS POINTS

The module should work with any standard access point. Roving Networks has tested the module with the following access points:

- Cisco Aeronet series
- Linksys (both standard and open WRT Linux)
- Netgear WGR614 v8
- Netgear WGN54
- DLINK dir-615
- Airlink 101
- Apple Airport express
- Buffalo networks
- Ad hoc mode (Apple iPhone, Microsoft Windows PC with XP, Vista, Ubuntu Linux)

Access points that are set to MIXED mode (WPA1 and WPA2) may cause problems during association because some of them incorrectly report their security mode.

Roving Networks currently does not support WPA2-Enterprise (radius server authentication, EAP-TLS)

COMMAND LIST

Tables 32 through 36 provide a listing of all available commands and their defaults. For more detailed information refer to "Command Reference" on page 5.

Table 32. Set Command List

Command	Default	Description
set adhoc beacon <*value*>	100	Sets the ad hoc beacon interval in milliseconds.
set adhoc probe <*value*>	5	Sets the ad hoc probe timeout in seconds.
set broadcast address <*address*>	255.255.255.255	Sets the address to which the UDP hello/heartbeat message is sent.
set broadcast interval <*mask*>	7	Sets the interval (in seconds) at which the hello/heartbeat UDP message is sent.
set broadcast port <*value*>	55555	Sets the port to which the UDP hello/heartbeat message is sent.
set comm $ <*char*>	$	Sets character used to enter command mode to <*char*>.
set comm close <*string*>	*CLOS*	Sets the ASCI string that is sent to the local UART when the TCP port is closed.
set comm open <*string*>	*OPEN*	Sets the ASCI string that is sent to the local UART when the TCP port is opened.
set comm remote <*string*>	*HELLO*	Sets the ASCI string that is sent to the remote TCP client when the TCP port is opened.
set comm idle <*value*>	0	Sets the idle timer value in seconds.
set comm match <*value*> | <*hex*>	0	Sets the match character in hex or decimal.
set comm size <*value*>	64	Sets the flush size in bytes.
set comm time <*value*>	10	Sets the flush timer.
set dns address <*address*>	0.0.0.0	Sets the IP address of the DNS sever.
set dns name <*string*>	server1	Sets the name of the host for TCP/IP connections to <*string*>.
set dns backup <*string*>	rn.microchip.com	Sets the name of the backup host for TCP/IP connections to <*string*>.
set ftp addr <*address*>	0.0.0.0	Sets the FTP server's IP address of the FTP server.
set ftp dir <*string*>	public	Sets the starting directory on the FTP server.
set ftp filename <*filename*>	See description	Sets the name of the file that is transferred when issuing the **ftp u** command, where <*filename*> is the firmware image. Default is wifly_GSX-<*version*>.img (RN-131) wifly_EZX-<*version*>.img (RN-171).
set ftp mode <*mask*>	0x0	Sets the ftp mode, where <*mask*> indicates active or passive mode. Default is passive.
set ftp remote <*value*>	21	Sets the FTP server's remote port number.
set ftp time <*value*>	200	Sets the FTP timeout value, where <*value*> is a decimal number that is five times the number of seconds required.
set ftp user <*string*>	roving	Sets the user name for accessing the FTP server.
set ftp pass <*string*>	Pass123	Sets the password for accessing the FTP server.
set ip address <*address*>	0.0.0.0	Sets the WiFly module's IP address.
set ip backup <*address*>	0.0.0.0	Sets a secondary host IP address.
set ip dhcp <*value*>	1	Enables/disables DHCP mode.
set ip flags <*mask*>	0x7	Sets the TCP/IP functions.

Command	Default	Description
set ip gateway *<address>*	0.0.0.0	Sets the gateway IP address.
set ip host *<address>*	0.0.0.0	Sets the remote host's IP address.
set ip localport *<value>*	2000	Sets the local port number.
set ip netmask *<address>*	255.255.255.0	Sets the network mask.
set ip protocol *<flag>*	2	Sets the IP protocol.
set ip remote *<value>*	2000	Sets the remote host port number.
set ip tcp-mode *<mask>*	0x0	Controls the TCP connect timers, DNS preferences, and remote configuration options.
set opt jointmr *<value>*	1000	Sets the join timer, which is the length of time (in ms) the join function waits for the access point to complete the association process.
set opt format *<flag>*	0x00	Sets the HTTP client/web server information.
set opt replace *<char>*	$ (0x24)	Sets the replacement character you use to indicate spaces in the SSID and pass phrases, where *<char>* is a single character.
set opt deviceid *<string>*	WiFly-GSX	Sets the configurable device ID.
set opt password *<string>*	"" (no password required)	Sets the TCP connection password.
set q sensor *<mask>*	0	Specifies which sensor pins to sample when sending data using the UDP broadcast packet or the HTTP auto sample function.
set q power *<value>*	0	Automatically turns on the sensor power.
set sys autoconn *<value>*	0	Sets the auto-connect timer in TCP mode.
set sys autosleep *<value>*	0	Sets the auto-sleep timer in UDP mode.
set sys iofunc *<mask>*	0x0	Sets the I/O port alternate functions.
set sys mask *<mask>*	0x20F0 (RN-131) 0x21F0 (RN-171)	Sets the I/O port direction.
set sys printlvl *<value>*	0x1	Controls the debug print messages printed by the WiFly module on the UART.
set sys output *<mask> <mask>*	None	sets the output GPIO pins high or low. The optional *<mask>* sets a subset of the pins.
set sys sleep *<value>*	0	Sets the sleep timer.
set sys trigger *<flag> or <mask>*	0x1	With this parameter setting, the module wakes from sleep state using the sensor input 0, 1, 2, and 3.
set sys value *<mask>*	0x0	Sets the default value of the GPIO pins' outputs upon power-up.
set sys wake *<value>*	0	Sets the automatic wake timer in seconds.
set time address *<address>*	129.6.15.28	Sets the time server address.
set time port *<value>*	123	Sets the time server port number.
set time enable *<value>*	0	Tells the module how often to fetch the time from the specified SNTP time server in minutes.
set time raw *<value>*	None	Allows you to set the RTC raw value from the console in seconds.
set uart baud *<value>*	9600	Sets the UART baud rate, where *<value>* is 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, or 921600.
set uart flow *<value>*	0	Sets the flow control mode and parity.
set uart instant *<value>*	Not applicable	Immediately changes the baud rate, where *<value>* is 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, or 921600.
set uart mode *<mask>*	0	Sets the UART mode register.
set uart raw *<value>*	Not applicable	Sets a raw UART value.
set uart tx *<value>*	Not applicable	Disables or enables the UART's TX pin (GPIO10), where *<value>* is 1 or 0.

Command	Default	Description
set wlan auth <*value*>	0	Sets the authentication mode.
set wlan channel <*value*> <*flag*>	0	Sets the WLAN channel, where <*value*> is a decimal number from 1 to 13 representing a fixed channel and <*flag*> is the optional character **i** (meaning immediate).
set wlan ext_antenna <*value*>	0	Determines which antenna is active, where <*value*> is 0 (use the chip antenna) or 1 (use the U.FL connector).
set wlan join <*value*>	1|0	Sets the policy for automatically associating with network access points.
set wlan hide <*value*>	0	Hides the WEP key and WPA passphrase, where <*value*> is 0 or 1.
set wlan key <*value*>	Not applicable	Sets the 128-bit WEP key, where <*value*> is EXACTLY 26 ASCII chars (13 bytes) in hex without the preceding 0x.
set wlan linkmon <*value*>	0 (disabled)	Sets the link monitor timeout threshold, where <*value*> is a decimal number representing the number of failed scans before the module declares **AP is Lost** and de-authenticates.
set wlan mask <*mask*>	0x1FFF (all channels)	Sets the WLAN channel mask, which is used for scanning channels with auto-join policy 1 or 2).
set wlan phrase <*string*>	rubygirl	Sets the passphrase for WPA and WPA2 security modes.
set wlan rate <*value*>	12	Sets the wireless data rate.
set wlan ssid <*string*>	roving1	Sets the SSID with which the module associates.
set wlan tx <*value*>	0	Sets the Wi-Fi transmit power, where <*value*> is a decimal number from 1 to 12 that corresponds to 1 to 12 dBm.

Table 33. get Commands

Command	Description
get adhoc	Displays the ad hoc settings.
get broadcast	Displays the broadcast UPD address, port, and interval.
get com	Displays the communication settings.
get dns	Displays the DNS settings.
get everything	Displays all of the configuration settings, which is useful for debugging.
get ftp	Displays the FTP settings.
get ip <*char*>	Displays the IP address and port number settings, where <*char*> is the optional parameter **a**. Using <*char*> returns the current IP address.
get mac	Displays the device's MAC address.
get option	Displays the optional settings such as the device ID.
get sys	Displays the system settings, sleep and wake timers, etc.
get time	Displays the time server UDP address and port number.
get wlan	Displays the SSID, channel, and other WLAN settings.
get uart	Displays the UART settings.
ver	Displays the firmware version.

Table 34. Status Commands

Command	Description
show battery	Displays current battery voltage, and is only applicable to Roving Networks' battery-powered products such as the RN-370 and temperature sensors (ISENSOR-CB).
show connection	Displays the connection status in the hex format 8<*XYZ*>.
show io	Displays the GPIO pins' level status in the hex format 8<*ABC*>.
show net <*char*>	Displays the current network status, association, authentication, etc., where <*char*> is the optional parameter **n**. Using the **n** parameter displays only the MAC address of the access point with which the module is currently associated.
show q <*value*>	Displays the value of the analog interface pin, where <*value*> is 0 to 7.
show q 0x1<*mask*>	Displays multiple analog interface values simultaneously.
show rssi	Displays the last received signal strength.
show stats	Displays the current statistics, packet RX/TX counters, etc.
show time	Displays the number of seconds since the module was last powered up or rebooted.

Table 35. Action Commands

Command	Description
$$$	Use this command to enter command mode.
close	Disconnects a TCP connection.
exit	Exits command mode.
factory RESET	Loads the factory defaults into the module's RAM and writes the settings to the standard configuration file. You must type the word **RESET** in capital letters.
join <*string*>	Instructs the WiFly module to join the network indicated by <*string*>.
join # <*value*>	Use this command to join a network that is shown in the scan list, where <*value*> is the entry number listed for the network in the scan list.
leave	Disconnects the module from the access point to which it is currently associated.
lites	Causes the LEDs on the RN-134 evaluation board and RN-370 WiFly serial adapter to blink. Using this command a second time stops the blinking.
lookup <*string*>	Causes the module to perform a DNS query for host name <*string*>.
open <*address*> <*value*>	Opens a TCP connection to <*address*>, where <*value*> is the port number.
ping <*string*> <*value*>	Pings a remote host, where <*string*> is a parameter setting and <*value*> is the number of pings. The default is 1 packet.
reboot	Forces the module to reboot (similar to a power cycle).
scan <*value*> <*char*>	Performs an active probe scan of access points on all 13 channels. The default is 200 ms/channel.
sleep	Puts the module to sleep.
time	Sets the real-time clock by synchronizing with the time server specified with the time server (**set time**) parameters.

Table 36. File I/O Commands

Command	Description
del <*string*> <*value*>	Deletes a file.
load <*string*>	Reads in a new configuration file.
ls	Displays the files in the system.
save <*string*>	Saves the your configuration settings to a file.
boot image <*value*>	Makes a file represented by <*value*> the new boot image.
ftp update <*string*>	Deletes the backup image file, retrieves a new image file, and updates the boot pointer to the new image.

KNOWN PROBLEMS

The firmware has the following known issues:

- *Flow control*—RTS may fail to de-assert quickly enough for some high-speed CPUs to stop sending data bytes correctly. For high-speed transfers at baud rates greater than 460,800, Roving Networks recommends limiting the RX data to the maximum Ethernet frame (1,460 bytes) and using a protocol to acknowledge that the remote host receives the data.

CURRENT FIRMWARE FEATURES & FIXES

VERSION 2.36/2.45 9/14/2012

- Firmware versions 2.36 and 2.45 are being shipped together. Version 2.36 supports ad hoc mode, while version 2.45 supports soft AP mode. All modules shipped with these firmware versions will run version 2.36 by default to maintain backward compatibility with previous firmware versions.

  You can change the firmware image using the **boot image** <*value*> command. After you change the boot image, you MUST reset the module back to the factory defaults using the **factory RESET** and **reboot** commands.

  - **Wifly_EZX-236.img**—Ad hoc mode firmware for RN-171
    Wifly_GSX-236.img—Ad hoc mode firmware for RN-131
  - **Wifly_EZX-245.img**—Soft AP mode for RN-171
    Wifly_GSX-245.img—Soft AP mode for RN-131
- In firmware version 2.36 (ad hoc version), the auto join feature is enabled to maintain backwards compatibility. In version 2.45, auto join is disabled and you must explicitly enable auto join mode using the **set wlan join 1** command.
- The firmware now supports parity with the **set uart flow** command.
- Added the **i** flag to the **set wlan channel** command, which changes the channel immediately. You can use this feature to go into AP mode without having to reboot or save the settings.
- In AP mode, de-authentication with the link monitor closes the TCP connection, flushes the UART buffer, and tries to clear stuck RTS flow control.
- In some cases flow control can get “stuck,” e.g., during a tcp_close or de-authentication in which the UART cannot transmit a TCP packet and is holding it. Added a fix to attempt to clear the buffer.
- In previous firmware, the associated status is always set even if no clients are joined. In version 2.45, the red LED correctly shows the status if there are no devices joined. If you use ALTERNATE IO for associated, it is high if there are 1 or more clients, and low if there are no clients.
- In AP mode, the module supports 7 connections (DHCP and AP).
- Fixed an issue with the ping command.

VERSION 2.30 10/26/2011

- Added support for incorrect WPA modes, namely WAPv1 with AES encryption and WPAv2 with TKIP encryption.
- Added support for WEP shared mode
- Increased FTP filename size to 64 bytes
- Added a new reboot register in ad hoc parameters. This register is reserved for future development and should not be used. Please leave it to default values.
- Added a new debug register in the system parameters. This register is reserved for future development and should not be used. Please leave it to default values.

VERSION 2.27 09/08/11

- Added support for Wi-Fi Protected Setup (WPS) push button mode
- Added support for WEP64 encryption
- Added support for backup IP address
- Added a new TCPMODE register to control TCP connect timers, force DNS and remote configuration
- Added new bit in UART mode register (bit 5) in which replaces the <2.23>\r\n in console with the replace character
- Fixed the FTP file “put” mode so it over rides HTTP mode. In version 2.23 if HTTP protocol is set and/or option format is set, extra data would be added to FTP put file data. This has been fixed in version 2.28.
- Fixed a bug where if the TCP_CLIENT mode is set, the module would randomly attempt outgoing connections.
- Fixed a bug in FTP data write mode whereby sometimes the *OPEN* status string came back over the UART before the file transfer was actually ready. This fix also improves the speed of FTP open in write mode, such that the *OPEN* will be faster.

VERSION 2.23 04/03/2011

- Created new set of sensor commands: **set q sensor** <*mask*> and **set q sensor** <*value*>. Also, sensor power can now be configured and applied either only when sampling of sensor inputs occur or at power up and removed upon power down and sleep.
- Added a new FTP client mode to get and put files to a FTP server. Files retrieved from the server are sent over the UART and files created from the UART are stored on the FTP server.
- A new scan output format is implemented in addition to the default output format. This new microprocessor friendly format makes string processing of the scan output very easy.
- A new UART heartbeat feature is implemented to notify the embedded microprocessor of the state of the WiFly module. The heartbeat message is sent over the UART when the WiFly module is in data mode and not connected to any remote host.

- Fixed a bug in the **set uart instant** <*value*> command wherein the WiFly module would not return an AOK over telnet. Now when this command is issued, it returns an AOK over telnet and **does not** exit command mode.
- Enabled scanning for wireless networks remotely when in ad hoc mode. When the **scan** command is issued, ad hoc is temporarily disabled and results of the scan are sent over telnet.
- The behavior of the auto connect timer has changed.
- Added the ability to set RTC from console.
- Added a feature wherein the module can be put to sleep using GPIO8.

VERSION 2.21 07/11/2010

- The firmware checksum the image (and compare to the stored values in the file) now before committing it to flash and updating the boot record after download. If the checksum fails firmware prints "UPDATE FAILED" and deletes the image.

VERSION 2.20 06/14/2010

Fixes

- Passphrase is now accepts up to 64 chars. A bug introduced in 2.19 causes the wlan passphrase to be truncated to 32 characters (making it impossible to enter a 32 byte HEX literal PSK).
- Fixed DHCP status when link to Access Point (AP) is lost. It was still reporting DHCP OK. It is now cleared and new DHCP session will start once AP link is reestablished).
- Fixed a bug whereby UDP receive becomes disabled (no packets are received) if AP-LOST and then re-established.
- Improved handling of AP disconnect, and AP link lost due to linkmon timeout or other disconnect.
- If TCP connection was active, connection could be in hung/incorrect state, and once AP is regained in some cases this would not recover. This has been fixed in this version. Refer section **set ip flags** <*value*> for more information.
- Added new setting to the UART mode **set uart mode 0x10**.

Features

- Disabled the auto-join feature when in command mode. Auto-join causes WiFly to become unresponsive to $$$ or commands during The period when auto-joining, when auto-joining is failing do to non-existent AP, making it hard to process or interpret commands. Once command mode is exited, auto join will re-enable.
- There are new levels of print out diagnostics that can be enabled/disable with the **sys print** variable.
- Ability to add prefix to HTML client post, specifically the ability to append **&id=** and **&rtc=** in the HTML message.

Roving Networks, Inc.
102 Cooper Court
Los Gatos, CA 95032
+1 (408) 395-5300
www.rovingnetworks.com