WIFLY GSX

802.11 b/g *wireless LAN Modules*

User Manual and Command Reference

RN-131G, RN-131C, RN-134, RN-121, RN-123 & RN-125

Version 2.21
July 11th , 2010

1. Overview

The "WiFly" radio module is a complete standalone embedded wireless LAN access device. The device has on board TCP/IP stack and applications. Requiring only 4 pins (POWER, TX, RX, GND) to design in. Once initial configuration is set, the radio can automatically access the Wi-Fi network and send/receive serial data over UART.

- Fully Qualified and Wi-Fi Certified 2.4GHz IEEE 802.11b/g transceiver
- High throughput, up to 4Mbps sustained data rate with TCP/IP and WPA2
- Ultra-low power (4uA sleep, 40mA Rx, 210mA max Tx)
- Small, compact surface mount module
- On board ceramic chip antenna and U.FL connector for external antenna
- 8 Mbit flash memory and 128 KB RAM
- UART and SPI (future) data/control interfaces
- 10 general purpose digital I/O
- 8 analog inputs
- Real-time clock for wakeup and time stamping/data logging
- Accepts 3.3V regulated or 2-3V battery with on board boost regulators
- Supports Adhoc and Infrastructure mode connections
- On board ECOS-OS, TCP/IP stacks
- Wi-Fi Alliance certified for WPA2-PSK
- FCC / CE/ ICS certified and RoHS compliant

Features

- Host Data Rate up to 1 Mbps for UART, 4Mbps SPI
- Memory 128 KB RAM, 2 MB ROM, 2 KB battery-backed memory, 8 Mbit Flash.
- Intelligent, built-in power management with programmable wakeup
- Can be powered from regulated 3.3VDC source or 2.0-3.0V batteries

- Real time clock for time stamping, auto-sleep and auto-wakeup modes
- Configuration over UART or wireless interfaces using simple ASCII commands
- Over the air firmware upgrade (FTP), and data file upload.
- Secure Wi-Fi authentication WEP-128, WPA-PSK (TKIP), WPA2-PSK (AES).
- Built in networking applications DHCP client, UDP, DNS client , ARP, ICMP ping, FTP, TELNET, HTTP
- 802.11 power save and roaming functions

2. Hardware Interface

See the specific module data sheets on the Roving Networks website for hardware specifications, and layout information.

2.1. Power

There are two options for powering the RN-131G module directly.

DC SUPPLY:

Apply 3.3 VDC power to VBATT (pin 20), and V3.3IN (pin 21).
Tie 3.3VREG-IN (pin 18) to GROUND.
Leave 3.3V-REG-OUT (Pin 17) floating/no connect.

BATTERY:

Apply battery = 2.0 to 3.3VDC to VBATT (pin 20).
Leave V3.3IN pin 21 floating/no connect.
Tie pin 17 to pin 18. (This enables the on board battery boost 3.3V switcher).

There is a built-in brownout monitor which will shut down the chip when the voltage drops below 2.0 VDC.

WARNING: Do NOT exceed the voltage ratings damage to the module will result.

#1: The Sensor inputs SENS0-7 are extremely sensitive to over voltage. Under no conditions should these pins be driven above 1.2VDC. Placing any voltage above this will permanently damage the radio module and render it useless.

#2: Placing 5VDC or any voltage above 3.3Vdc into the VDD pins of the module will permanently damage the radio module.

#3: Placing 3.3Vdc into the PIO's while they are set as outputs will permanently damage the module. The failure mode is a short across GND and VCC.

2.2. Reset

Reset is active LOW and is optional/does not need to be connected. The reset pin is 3.3V tolerant and has an internal pull up of 100K to the VBATT.

2.3. UART

Connect a common ground when using the external TX, RX inputs.

For a 3 wire DB-9 interface (connect TX, RX, GND only)

Factory default is hardware flow control disabled; CTS and RTS are not required.

PIO's are not 5.0 VDC tolerant. If using a 5.0 VDC circuit, input, PIO and UART input pins require a resistor divider. A suggestion is to use a 10K resistor in series with 20k resistor to ground.

2.4. Status Indictors

PIO 4, 5 and 6 are active high and can be connected to external LEDs to provide network, connection and data status.

Condition	PIO6=Red LED	PIO5=Yellow LED	PIO4=Green LED
ON solid			Connected over TCP
Fast blink	Not Associated	Rx/Tx data transfer	No IP address
Slow blink			IP address OK
OFF	Associated		

3. Configuration

3.1. Entering Command Mode

Upon power up, the device will be in data mode. To enter command mode, exactly the three characters **$$$** must be sent. The device will respond with **CMD**.

While in command mode, the device will accept ASCII bytes as commands.
To exit command mode, send **exit<cr>.** The device will respond with "EXIT".

Parameters, such as the SSID, channel, IP address, Serial Port settings, and all other settings can be viewed and configured in command mode.

ASCII characters can be sent through a terminal emulator connected to the UART or via Telnet. When using the UART communications settings should match the settings used when RN-131g connects, for example: the default is 9600 baudrate, 8 bits, No Parity, 1 stop bit, and hardware flow control disabled.

Use TeraTerm as your terminal emulator. Please **DO NOT** use HyperTerminal as it is known to have issues with our products. TeraTerm can be downloaded from our website:
http://www.rovingnetworks.com/support/teraterm.zip.

Type **$$$** on in the terminal emulator. You should see "**CMD**" returned to you. This will verify that your cable and comm. settings are correct. Most valid commands will return an "**AOK**", response, and invalid ones will return an **"ERR"** description.

To exit command mode, type **"exit"<cr>.**

NOTE: You can enter command mode locally over the UART interface at any time when not connected, and also when connected if the appropriate settings are enabled.

NOTE: When the WiFly GSX module is powered up, it tries to auto associate to the Access Point stored in the config settings. If for some reason the module cannot find the Access Point, it goes into auto association mode and gets busy scanning and trying to join a network. This may cause the UART to become unresponsive for a brief amount of time and you may lose the data sent to the module while the module is in this "not associated" state making it difficult to get into command mode and configure the module

Version 2.21 of the firmware fixes this issue. The auto-join feature is disabled when in command mode. This makes it easy to configure the module. Auto-join will re-enable when you exit out of command mode.

The auto join feature can be disabled by setting the **set wlan join 0**. This will prevent the WiFly GSX module to attempt to associate to a network that does not exist.

Another alternative is to boot the module in adhoc mode by using the PIO9 adhoc/factory reset jumper. If this is high on power up, the module will not associate to any network; it will use the temporary adhoc mode. When in adhoc mode, you can configure the network settings.

Remote configuration using ADHOC mode

Using adhoc mode to configure the device eliminates the need for the module to be associated with a network access point. In adhoc mode the module creates it own "on demand" network that you can connect to via your computer like you would to any other network.

To enable adhoc mode via hardware set *PIO9* high (3.3V) at power up. On the RN-134 PIO9 is on the J1 jumper block. When the module powers up with PIO9 set high, the WiFly module creates an adhoc network with the following

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 adhoc jumper in place the above settings override the current saved configuration settings.

From your computer, connect to the WiFly-GSX-XX network. This is an open network which does not require a pass phrase or pass key. Note: currently the WiFly only supports OPEN mode for creating adhoc networks.

NOTE: It may take a couple of minutes for Auto IP in Windows to assign an IP address and connect to the network. You can check IP address of your Windows computer by running the ***ipconfig*** command in the command window. If connected, this command will show you the IP address and net mask for your computer.

The IP address assigned by Auto IP must be on the subnet 169.254.x.y otherwise the WiFly GSX module will not be accessible.

NOTE: If your machine has both wireless and wired interface hardware you will need to disable the wired LAN interface hardware before connecting to the adhoc 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 and you have a good IP address, telnet into the WiFly module on port 2000
telnet 169.254.1.1 2000
You should see the response "*HELLO*"

You can now enter command mode and configure the module.

4. WiFly Command Reference

4.1. Command Syntax

Commands begin with a keyword, and have optional additional parameters, generally space delimited. Commands and options **are** case sensitive. Hex input data can be upper or lower case. String text data, such as SSID is also case sensitive.

The first command is fully decoded and must be complete. Other command parameters can be shorted by using only the first character.

For example,

set uart baudrate 115200 is valid,

set uart b 115200 is also valid,

set u b 115200 is also valid, however,

s uart baudrate 115200 is **NOT** valid.

Numbers can be entered as either decimal, (like 115200 above) or HEX. To enter HEX, use **0x<value>.** For example, the HEX value FF would be entered as 0xFF.

4.2. Command Organization

Commands fall into 5 general categories:

SET COMMANDS	Take effect immediately, permanently (save command issued).
GET COMMANDS	Retrieve the permanently stored information for display to user.
STATUS COMMANDS	See what is going on with the interface, IP status, etc.
ACTION COMMANDS	Perform action such as scan, connect, disconnect, etc.
FILE IO COMMANDS	Upgrade, load and save configuration, delete files, etc.

NOTE: You must save any changes made 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 file called "config". The set commands actually only modify the RAM copy of variables in the system. In general, the IP, WLAN and UART settings need a save and reboot to take effect, since they operate at boot up time. For example you only associate, set the channel and get your ip address once at power up.

Most of the other commands take effect immediately like the COMM settings and timers. This allows temporary change of parameters "on the fly" to test features, minimizes power usage and saves on flash re-write cycles.

Once all configuration is complete, the user must save the settings using the **save** command to store the configuration data, otherwise it will not take effect upon reboot or reset. Multiple configurations can be stored by using the **save <filename>** command, and these configurations can be loaded using the **load <filename>** command.

5. SET Commands

These commands begin with "set". There are 6 major categories.

Adhoc	controls the adhoc parameters
Broadcast	controls the broadcast hello/heartbeat UDP message
COMM	communication and data transfer, timers, matching characters
DNS	DNS host and domain
FTP	FTP host address and login information
IP	IP settings
Option	optional and not frequently used parameters
Sys	system settings such as sleep and wake timers
Time	timer server settings
UART	serial port settings such as baudrate and parity
WLAN	wireless interface settings, such as ssid, chan, and security options

5.1. Adhoc Parameters

set adhoc beacon <ms> sets the adhoc beacon interval in milliseconds. Default is 100.

set adhoc probe <num> sets the adhoc probe timeout in seconds. Default is 60. This is the number of seconds waiting for probe responses before declaring "ADHOC is lost" and disabling the network interface.

5.2. Broadcast Parameters

set broadcast address <addr> sets the address to which the UDP hello/heartbeat message is sent. The default address is 255.255.255.255

set broadcast interval <value> sets the interval at which the hello/heartbeat UDP message is sent. Interval is specified in seconds. The value is a mask that is compared to a free running seconds counter. For example if interval = 0x7, a packet will be sent every 8 seconds. The minimum interval value is 1 (every 2 seconds) and max value is 0xff (every 256 seconds). Setting the interval value to zero disables sending UDP broadcast messages. The default interval is 7.

set broadcast port <port> sets the port number to which the UDP hello/heartbeat message is sent. The default port is 55555.

5.3. COMM Parameters

set comm $ <char> sets character used to enter command mode. Typically used when "$$$" is a possible data string. Default is '$'. Care should be taken when setting this to note the new character as once this setting is saved every subsequent reboot will ignore "$$$" and look for "<char><char><char>".

set comm close <string> sets the ASCI string that is sent to the local UART when the TCP port is closed. If no string is desired, use 0 as the <string> parameter. Max string length is 32 characters. Default is *CLOS*

set comm open <string> sets the string that is sent to the local UART when the TCP port is opened. If no string is desired, use 0 as the <string> parameter. Max string length is 32 characters. Default is *OPEN*

set comm remote <string> sets the string that is sent to the remote TCP client when the TCP port is opened. If no string is desired, use 0 as the <string> parameter. Max string length is 32 characters. Default is *HELLO*

set comm idle <secs> sets the Idle Timer Value. This is the number of seconds with no transmit or receive data before the connection is closed automatically. Default is 0, never disconnect on idle.

set comm match <value> sets match character. An IP packet will be sent each time the match character appears in the data. Value is entered as the decimal (13) or hex (0xd) of the of the ASCII character. Default is 0, disabled. The match character is one of three ways to control TCP/IP packet forwarding. The others are size and timer. For more information see section 10.1 on System Timers and Auto Connect Timers and section 10.4 on UART Receiver.

set comm size <value> sets the flush size. An IP packet will be sent each time "value" bytes are received. Default is 64 bytes. You should set this value to the largest

possible setting to maximize TCP/IP performance. Maximum value = 1420 (at 9600) bytes.

NOTE: This value is set automatically when the baudrate is set, in an attempt to optimize the link. It is assumed that higher baudrates equates to more data and hence the flush size is increased.

Flush size is one of three ways to control TCP/IP packet forwarding. The others are match character and timer. For more information see section 10.4 on UART Receiver.

set comm time <num> sets the flush timer. An IP packet will be sent if no additional bytes are received for "num" milliseconds. Num is one milliseconds interval. 1 is the minimum value. Default is 10 (10 milliseconds). Setting this value to 0 will disable forwarding based on the flush timer.

Flush timer is one of three ways to control TCP/IP packet forwarding. The others are match character and size. For more information see section 10.1 on System Timers and Auto Connect Timers

5.4. DNS Parameters

set dns address <addr> sets the IP address of the DNS sever. This is auto-set when using DHCP, and needs to be set in STATIC IP or Auto-IP modes.

set dns name <string> sets the name of the host for TCP/IP connections.

set dns backup <string> sets the name of the backup host for TCP/IP connections.

5.5. FTP Parameters

set ftp filename <file> sets the name of the file transferred when issuing the "ftp u" or "ftp g" commands.

set ftp addr <addr> sets the ftp server IP address.

set ftp remote <port> sets the ftp server remote port number (default is 21).

set ftp user <name> sets the ftp user name for accessing the FTP server.

set ftp pass <pass> sets the ftp password for accessing the FTP server.

5.6. IP Parameters

set ip address <addr>

sets the IP address of the WiFly GSX module. If DHCP is turned on, the IP address is assigned and overwritten during association with the access point. IP addresses are "." delimited. Note this is different from the RN-111b module which is space delimited!

Example: "set ip a 10.20.20.1"

set ip backup <addr>

sets a secondary host IP address. If the primary host IP is not reachable the module will try the secondary IP address if set.

set ip dchp <value>

enable/disable DHCP mode. If enabled, the IP address, gateway, netmask, and DNS server are requested and set upon association with access point. Any current IP values are overwritten.

DHCP Cache mode can reduce the time it takes the module to wake from deep sleep thus saving power. In cache mode, the lease time is checked and if not expired, the module uses the previous IP settings. If the lease has expired the module will attempt to associated and use DHCP to get the IP settings. DHCP cached IP address does not survive a power cycle or reset.

Mode	Protocol
0	DHCP OFF, use stored static IP address
1	DHCP ON, get IP address and gateway from AP
2	Auto-IP, generally used with Adhoc networks
3	DHCP cache mode, Uses previous IP address if lease is not expired (lease survives reboot)
4	Reserved for future use

set ip flags <value>

Set TCP/IP functions. Value is a bit mapped register. Default = 0x7.

Bit	Function
0	TCP connection status. See note below
1	Bypass Nagle algorithm and use TCP_NODELAY
2	TCP retry enabled (42 total)
3	UDP RETRY (attempts retry if no ACK from UDP)
4	DNS host address caching enabled
5	ARP table caching enabled
6	UDP auto pairing enabled
7	Add 8 byte timestamp to UDP or TCP packets

NOTE: When the link to an associated to an access point is lost while a TCP connection is active, the TCP connection can be left in hung/ inconsistent state. In some cases, the TCP connection will not recover. In version 2.20 and later, if the link to the access point is regained within 60 seconds, the TCP connection will survive.

With version 2.20 we have changed the operation of bit0 in the "ip flags" register. Previously this bit specified the TCP copy function, but controls the TCP socket function while associated on a network.

- If bit 0 is set (default) TCP connections are kept open when the connection to the access point is lost.
- If bit 0 is cleared (by setting "set ip flags 0x6" for example) then when the connection to the access point is lost and TCP is connected, the connection will be closed.

set ip gateway <addr> sets the gateway IP address, If DHCP is turned on, the gateway IP address is assign and overwritten during association with the access point.

set ip host <addr> sets the remote host IP address. This command is used for making connections from the WiFly module to a TCP/IP server at the IP address <addr>.

set ip localport <num> sets the local port number.

set ip netmask <value> sets the network mask. If DHCP is turned on, the net mask is assign and overwritten during association with the access point.

set ip protocol <value> sets the IP protocol. Value is a bit mapped setting. To connect to the WiFly GSX module over TCP/IP such as Telnet the device must have the use the TCP Server protocol / bit 2 set. To accept both TCP and UDP use value = 3 (bit 1 and bit 2 set)

Bit Position	Protocol
0	UDP
1	TCP Server & Client (Default)
2	Secure (only receive packets with IP address matches the store host IP)
3	TCP Client only
4	HTTP client mode

set ip remote <value> sets the remote host port number.

5.7. Optional Parameters

set opt jointmr <msecs> Join timer is the time in milliseconds (default=1000) the join function will wait for the an access point to complete the association process. This timer is also the timeout for the WPA handshaking process.

set option sensor <mask> Bitmask value that determines which sensor pins to sample when sending data using the UDP broadcast packet and the HTTP auto sample function.

set opt format <mask> settings for HTTP client/web server value is a bitmapped register. Refer to section 13 for more details.

Bit	Function
0	Automatically send HTML data header based on broadcast interval.
1	Send users BINARY data (converted to ASCII hex)
2	Sample the GPIO and AtoD pins and format to ASCII hex
3	Appends **&id= <**the value of the deviceid string set with "set opt device <string>">
4	Appends &rtc= <real time clock value in message as 32 bit HEX value in format aabbccddeeff>

set opt replace <char> replacement character for spaces. The replacement character is used when entering SSID and pass phrases that include space. This is used by the WiFly GSX command parser only. Each occurrence of the replacement character is changed into a space. The default is "$" (0x24)

set opt deviceid <string> Configurable Device ID - can be used for storing serial numbers, product name or other device information. This information is sent as part of the broadcast hello packet that is sent as a UDP. The current value can be shown with the "get option" or "show deviceid" commands. Max string size is 32 bytes. The default is "WiFly-GSX".

set opt password <string> TCP connection password. Provides minimal authentication by requiring any remote device that connects to send and match a challenge <string>. When set, all newly opened connections must first send the exact characters that match the stored password otherwise the WiFly module will close the connection. When the password is set the WiFly module sends the string "PASS?" to the remote host. All characters in the string must be sent in one TCP packet. Max string size is 32 bytes. To disable the password feature use string=0 which is the default.

5.8. System Parameters

set sys autoconn <secs> TCP mode: sets the auto connect timer. This command causes the module periodically connect to the host. The timer <secs> determines how often to connect to the stored remote host. If set to 1, the module will only make one attempt to auto connect upon power up. If set to 2 or greater auto connect will re-open the connection after the connection is closed. Default=0 disables.

set sys autosleep <num> Sets the auto-sleep timer. 0 disables. If the protocol is set to UDP ONLY, this timer is used as a quick sleep function. Device will sleep <num> ms after transmission of the first UDP packet.

set sys iofunc <value> sets the IO port alternate functions. Bit-mapped value. For more details see section 10.5

set sys mask <mask> sets the IO port direction mask. Bit-mapped value. For more information see section 10.5

set sys printlvl <value> sets numerous print functions. 0 = quiet, 1 = connect information Default is 1 . Please refer section 10.6 on Setting Debug Print levels

set sys output <value> <mask> sets output PIO pins to HIGH or LOW. Bit-mapped value. Optional mask only sets a subset of pins.

set sys sleep <secs> sets the sleep timer. 0 disables.

NOTE: If not using Sensor pins to wake the module, be sure to set the wake timer before issuing the sleep timer or the module will not wake up.

See section 10.1 for more details on using system timers

set sys trigger <value> sets the sensor input(s) to wake on (0-3). Bit-mapped value. 0 disables.

set sys wake <secs> sets the auto wake timer. 0 disables. See section 10.1 for more details on using system timers

5.9. Time Server Parameters

set time address <addr> sets the time server address. (sNTP servers)

set time port <num> sets the time server port number. Defaults to 123 which is almost always the sNTP server port.

set time enable <value> Enable or disable fetching time from the specified sNTP time server. Default=0= disabled. A value or 1 gets time only once on power up. Any value > 1 gets time continuously every <value> minutes.

5.10. UART Parameters

set uart baud <rate> set the UART baud rate. Valid settings are {2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600}.

Example : "set u b 9600" sets the baud rate to 9600 baud.

NOTE: the RS232 interface on the RN-134 does not work above 230400

set uart instant <rate> This immediately changes the baudrate. This is useful when testing baudrate settings, or switching baudrate "on the fly" remotely while connected over TCP. This setting does not affect configuration. Returns the AOK response, and then this command will exit command mode.

set uart raw <rate> sets a RAW UART value. Used to set non-standard rates. The lowest possible baud rate is 2400.

Example : "set u r 7200" sets the baud rate to 7200 baud.

set uart flow <0,1> sets the flow control mode. Default=0=off, 1= hardware RTS/CTS. **NOTE: once flow control is enabled, it is important to properly Drive the CTS pin (active LOW enabled)** If CTS is HIGH, data will NOT be sent out the UART, and further configuration in command mode will be problematic as no response will be received.

set uart mode <mask> sets the UART mode register. This is a bit-mapped value.

Bit Position	Function
0	NOECHO - disables echo of RX data while in command mode
1	DATA TRIGGER makes connection on RX data
2	RAW mode (TCP stack disabled)
3	Enable Sleep on RX BREAK signal
4	UART RX data buffer. See note below for details*

***NOTE:** When a TCP connection is closed, currently if there is RX data in the UART receiver, it is held until

1) more chars come in, in which case it will get flushed, or
2) no chars come in and a new connection is made, then the chars will get forwarded.

If this setting is enabled (set uart mode 0x10), any unsent RX data is still in the buffer is flushed when a connection is closed.

set uart tx <0, 1> Disables or enables the TX pin= PIO10 of the UART. Disable will set PIO10 to an INPUT with weak pulldown.

NOTE: Due to an issue in the UART hardware, the UART does not support even or odd parity.

5.11. WLAN Parameters

set wlan auth <value> Sets the authentication mode. Not needed unless using auto join mode 2. i.e. ***set wlan join 2***

Note: During association the WiFly module interrogates the Access Point and automatically selects the authentication mode.

The current release of Wifly firmware supports these security modes:
• WEP-128 (open mode only, NOT shared mode)
• WPA2-PSK (AES only)
• WPA1-PSK (TKIP only)
• WPA-PSK mixed mode (some APs, not all are supported)

Value	Authentication Mode
0	Open (Default)
1	WEP-128
2	WPA1
3	Mixed WPA1 & WPA2-PSK
4	WPA2-PSK
5	Not Used
6	Adhoc, Join any Adhoc network

set wlan channel <value> sets the wlan channel, 1-13 is the valid range for a fixed channel. If 0 is set, then scan is performed, using the ssid, for all the channels set in the channel mask.

set wlan ext_antenna <0, 1> determines which antenna is active, use 0 for chip antenna, 1 for UF.L connector. Default = 0. Only one antenna is active at a time and the module must be power cycled after switching the antenna.

set wlan join <value> sets the policy for automatically joining/associating with network access points. This policy is used when the module powers up, including wake up from the sleep timer.

Value	Policy
0	Manual, do not try to join automatically
1	Try to join the access point that matches the stored SSID, passkey and channel. Channel can be set to 0 for scanning. (Default)
2	Join ANY access point with security matching the stored authentication mode. This ignores the stored SSID and searches for the access point with the strongest signal. The channels searched can be limited by setting the channel mask.
3	Reserved – Not used
4	Create an Adhoc network, using stored SSID, IP address and netmask. Channel MUST be set. DHCP should be 0 (static IP) or set to Auto-IP with this policy. (unless another Adhoc device can act as DHCP server) This policy is often used instead of the hardware jumper to creat a custom Adhoc network

set wlan hide <0, 1> Hides the WEP key and WPA passphrase. When set, displaying the wlan settings shows ****** for these fields. To unhide the passphrase or passkey, re-enter the key or passphrase using the set wlan key or set wlan passphrase command. Default = 0, don't hide.

set wlan key <value> sets the 128 bit WEP key. If you are using WPA or WPA2 you should enter a pass phrase with the set wlan passphase command. Key must be EXACTLY 13 bytes (26 ASCII chars). Data is expected in HEX format, "0x" should NOT be used here.

Example : "set w k 112233445566778899AABBCCDD"

Hex digits > 9 can be either upper or lower case.

The Wifly GSX only supports "open" key mode, 128 bit keys for WEP. WEP-128, shared mode is not supported as it is known to be easily compromised and has been deprecated from the WiFi standards.

set wlan linkmon <value> sets the link monitor timeout threshold. If set to 1 or more, WiFly will scan once per second for the AP it is associated with. The value is the threshold of failed scans before the WiFly declares "AP is Lost", de-authenticates. The WiFly will retry the association based on the join policy variable. A value of 5 is recommended, as some APs will not always respond to probes. Default is 0 (disabled). Without this feature, there is no way to detect an AP is no longer present until it becomes available again (if ever).

set wlan mask <value> sets the wlan channel mask used for scanning channels with the auto-join policy 1 or 2, used when the channel is set to 0. Value is a bit-map where bit 0 = channel 1. Input for this command can be entered in decimal or hex if prefixed with 0x. Default value is 0x1FFF (all channels)

set wlan num <value> sets the default WEP key to use. 1-4 is the valid range.

Example : "set w n 2" sets the default key to 2.

set wlan phrase <string> sets the passphrase for WPA and WPA2 security modes. 1-64 chars. The passphrase can be alpha and numeric, and is used along with the SSID to generate a unique 32 byte Pre-shared key (PSK), which is then hashed into a 256 bit number. Changing either the SSID or this value re-calculates and stores the PSK.

If exactly 64 chars are entered, it is assumed that this entry is already 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" would be entered "my$pass$word". The replacement character can be changed using the optional command **set opt replace <char>**.

Example : "set w p password" sets the phrase.

set wlan rate <value> sets the wireless data rate. Lowering the rate increases the effective range of the WiFly-GSX module. The value entered is mapped according to the following table

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

set wlan ssid <string> sets the wlan ssid to associate with. 1-32 chars.

NOTE: If the passphrase or ssid contain the SPACE (‘ ‘) characterS, these can be entered using substitution via the “$” character.

For example, if the ssid of the AP is “yellow brick road”
You would enter “yellow$brick$road”

Using the ‘get w” command will properly display the value:
SSID=yellow brick road.

set wlan window <value> sets the IP maximum buffer window size. Default is 1460 bytes.

6. Get Commands

These commands begin with “get”. They display the current values.

get adhoc	display all adhoc settings.
get broadcast	will display the broadcast UPD address, port and interval
get everything	displays all configuration settings, useful for debug.
get com	display comm. settings.
get dns	display DNS settings.
get ftp	display FTP settings.
get ip <a>	display IP address and port number settings. Optional parameter just returns the current IP address value.
get mac	display the device MAC address.
get option	display the option settings like device ID
get q	display the sensor settings (sensor mask and sensor power settings)
get sys	display system settings, sleep, wake timers, etc.
get time	display the time server UDP address and port number.
get wlan	display the ssid, chan, and other wlan settings.
get uart	display the UART settings.
ver	return the software release version

7. Status Commands

These commands begin with “show”, and they return the current values of variables in the system. In some cases, for example IP addresses, the current values are received from the network, and may not match the stored values.

show battery	Displays current battery voltage, (only valid for Roving battery powered product like the RN-370 and temperature sensors)

show connection Displays connection status in this HEX format: 8XYZ

Bit location	13-16	9-12	7	6	5	4	0-3
Function	fixed	channel	DNS found	DNS server	Authen	Assoc	TCP status
Value	8	1-13	1=resolved	1= contacted	1= OK	1=OK	0= Idle, 1=Connected 3= NOIP 4= Connecting

show io Displays IO pin levels status in this HEX format: 8ABC
Example: **show i** returns 8103 indicates pins 0, 1 and 9 high level.

show net <n> Displays current network status, association, authentication, etc. Optional parameter displays only the MAC address of the AP currently associated.

show rssi Displays current last received signal strength.

show stats Displays current statistics, packet rx/tx counters, etc.

show time Displays number of seconds since last powerup or reboot

show q <0-7> Display the value of the an analog interface pin from 0 to 7. The value returned will be in the format 8xxxxx where xxxxx is voltage in microvolts sampled on the channel you request with the 8 in front as a start marker.

show q 0x1<mask> Displays multiple analog interface values at once. The channels displayed is controlled by a bit mask which is proceeded by a 0x1xx where xx mask is the bit mask of the channels. For example, to read channels 0,1, and 7, send:

show q 0x183

Which returns 8<chan0>, 8<chan1>, 8<chan7>, \r\n

8. Action Commands

$$$ enter command mode Characters are PASSED until this exact sequence is seen. If any bytes are seen before these chars, or after these chars, in a 250ms window, command mode will not be entered and these bytes will be passed on to other side.

close disconnect a TCP connection.

exit	exit command mode. Exit command mode. "EXIT" will be displayed.
factory RESET	Loads factory defaults into the RAM configuration. **Note that the RESET must be capitalized**. This command also writes the settings out to the standard config file. After this command the module then needs to be rebooted for settings to take effect.
join <ssid>	joins the network <ssid>. If network is security enabled you must set the pass phrase with the **set wlan phrase** command prior to issuing the ***join*** command
join # <num>	join a network from the scan list. <num> is the entry number in the scan list that is returned from the scan command. If network is security enabled you must set the pass phrase with the **set wlan phrase** command prior to issuing the ***join*** command
leave	disconnects the module from the currently associated Access Point.
lookup <hostname>	performs a DNS query on the supplied hostname.
open <addr> <port>	opens a TCP connection to the given IP port and address. If no arguments are provided, the device will attempt to connect to the **stored** remote host IP address and remote port number. <addr> can also be a DNS hostname and will be resolved if entered.
ping <g | h | i | addr> <num>	ping remote host. Default sends 1 packet. Optional <num> sends <num> pings at 10 per second. ***Ping 10.20.20.12 10*** – pings IP address 10 times ***ping g*** pings the gateway, the gateway IP address is loaded if DHCP is turned on, otherwise it should be set with the ***set ip gateway <addr>*** command ***ping h*** pings the stored host IP address, the host IP address can be set with the ***set ip host <addr>*** command ***ping i*** pings a known Internet server at www.neelum.com by first resolving the URL (proves that DNS is working and proves the device has internet connectivity). ***ping 0*** terminates a ping command
reboot	forces a reboot of the device (similar to power cycle)

scan <time> <P>	Performs an active probe scan of access points on all 13 channels. Returns MAC address, signal strength, SSID name, security mode. Default scan time is 200ms / channel = about 3 seconds. **time is** an optional parameter, this is the time in ms per channel. For example, "scan 30" reduces the total scan time down to about 1 second. This command also works in Adhoc mode. If the optional **P** parameter is entered, the module will perform a passive scan, and list all APs that are seen in passive mode.
sleep	Puts the module to sleep mode. The module can come out of sleep mode by either sending characters over the uart or by using the wake timer.
time	Sets the Real time clock by synchronizing with the time server specified with the time server parameters (see section 5.9) This command sends a UDP time server request packet.

9. File IO Commands

del <name> <num>	Deletes a file. Optional <num> will override the name and use the sector number shown in the "ls" command.
load <name>	Reads in a new config file.
ls	Displays the files in the system
save	Saves the configuration to "config" (the default file).
save <name>	Saves the configuration data to a new file name
boot image <num>	Makes file <num> the new boot image.
ftp get <name>	Retrieves a file from the remote FTP server. If <name> not specified, the stored ftp filename is used.
ftp update <name>	Deletes the backup image, retrieves new image and updates the boot image.

10. Advanced features and Settings

This chapter describes the advanced features of the WiFly GSX module. It describes the techniques to put the module in sleep, wake up from sleep and methods to open a TCP connection when awake. We also discuss the uart flow control, alternative GPIO functions and Real Time Clock.

The table below describes the possible methods of putting the module to sleep.

Method	Interface	Description
sleep command	UART	Get into command mode using **$$$** and issue the **sleep** command
Sleep Timer	Internal RTC	Puts the module to sleep based on the **set sys sleep <secs>** command

To wake up the module from sleep, following options are available:

Method	Type	Description
Sensor Input **(1.2VDC ONLY)**	Sensor Pins	You can wake up the module on sensor pins 0-3 (**1.2V ONLY**). Use the **set sys trigger <value>** command to enable. Refer section 10.2 for details
Rx Pin **(1.2VDC ONLY)**	RX pin via Sensor 0	The RX pin on the RN-134 is tied to Sensor Pin 0. Use **set sys trigger 1** command to wake up on RX data. **NOTE**: You may drop the first byte of uart data. A better way is to wake up the module on CTS pin. Refer section 10.3 for details
CTS Pin **(3.3VDC ONLY)**	CTS pin via Sensor 1	The CTS pin on the RN-134 is tied to Sensor pin 1. Use **set sys trigger 2** command to wake up on CTS. Refer section 10.3 for details
Wake Timer	Internal RTC	Wakes up the module from sleep based on the **set sys wake <secs>** command
FORCE AWAKE	FORCE AWAKE pin	Input pulse of atleast 31µsecs duration (3.3V) will wake up the module.

When the module wakes up from sleep, it takes a certain amount of time (in milliseconds) to initialize the internal hardware. During this time, any data that is sent to the WiFly module over the uart will not be processed. You can monitor certain signals that indicate that the module is ready to accept data. These are described below.

Method	Interface	Description
RTS transition	RTS pin	Once the WiFly GSX module wakes up, the RTS line goes HIGH. Once the system is ready, the RTS is driven LOW. This can be monitored by the micro controller
Monitor GPIO 4	Alternative GPIO functions	Set the alternative functions for GPIO 4, 5 and 6 (Refer section 10.5.1). Once the module wakes up and connects to an AP, GPIO 4 goes high. This indicates the module is ready to receive data over the UART. Your micro controller can monitor GPIO 4

Once the module is awake, you can open a TCP connection to a remote host in a number of ways described below. The remote host can be set using the following commands:

set ip host <IP address> **OR** set dns name <string> // sets up the IP address OR URL of host
set ip remote <port number> // sets up the port number on which the host is listening
save // save settings in config file
reboot // reboots the module so that the settings take effect

Method	Type	Description
Auto connect	Internal RTC Timer	Connect out to the host at specific time intervals based upon the **set sys autoconn <secs>** command
Open	Uart	In command mode, you can 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 automatically connect out to host on uart data
GPIO 5	Alternative GPIO functions	Set the alternative functions for GPIO 4, 5 and 6 (Refer section 10.5.1). Set GPIO 5 HIGH to trigger TCP connection, LOW to disconnect

10.1. System Timers and Auto Connect Timers

The WiFly GSX module uses the Real Time clock to generate timers. The RTC is active even when the WiFly module is asleep. This makes it possible to put the module to sleep and wake up from sleep based on timer intervals using timers.

The WiFly module has the following timers available:

1. Sleep Timer: Used to put the WiFly module to sleep
2. Wake Timer: Used to wake the WiFly module from sleep
3. Auto-connect Timer: Used to automatically open a TCP connection
4. Idle Timer: Used to automatically close a TCP connection

There are 2 timers that can be used to put the module to sleep, and perform a wake up. If the sleep timer is enabled, the module will automatically go 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 the module is in COMMAND mode.

The sleep timer (which is the time the WiFly is awake) is a 32 bit number of seconds so it can be as high as 1.19 million hours.

The wake timer (which is the time the WiFly is asleep) is a 22 bit number of seconds so the maximum sleeping time is 1165 hours.

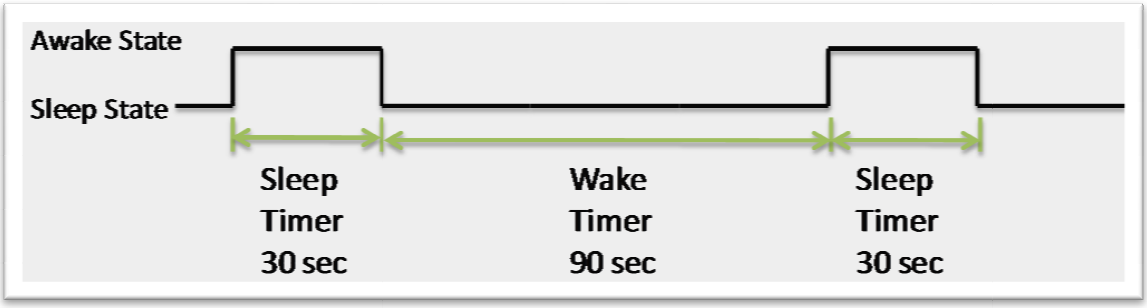
The sleep timer is set with : **set sys sleep <time>** time=decimal in seconds.

The wake timer will bring the module out of deep sleep.

The wake timer is set with: **set sys wake <time>** time=decimal in seconds.

For example, if you wanted 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 such

set wlan ssid my_net
set wlan passphrase my_pass
set sys sleep 30
set sys wake 90
save
reboot



The above diagram shows the transitions between the Sleep state and Awake state based on the sleep and wake timers.

10.1.1. UDP sleep and Connection timers

In UDP Only protocol mode (set ip proto 1), the autoconn timer is used as an auto-sleep timer.

Upon the start of transmission of the first UDP data packet this timer will count down, after which the module will go to sleep.

set sys autosleep <value> // UDP Only mode: sets the auto-sleep timer. Setting value=0 disables autosleep timer

The UDP auto-sleep timer is set using two variables. The timer interval is a product of the autosleep value and the comm flush timer (in milli seconds). The timer is decremented every "product" milliseconds.

For example, if you need a UDP sleep timer of 40 milli seconds, you need to set the following variables:

set sys autosleep 4 // Sets the autosleep value to 4
set comm timer 10 // Sets the comm timer to 10 ms (default value)

The resulting UDP sleep timer will be 4*10 ms = 40 ms. You can also use a combination of autosleep = 2 and comm timer = 20 ms to achieve the same effect.

Using a minimum value of 2 (when the default flushtime=10 ms) is recommended to ensure that the UDP packet gets transmitted. For larger packets the value should be increased.

10.1.2. TCP Connection Timers

Opening a TCP Connection:

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

set sys autoconn <secs>

This command causes the module periodically connect to the host. The timer <secs> determines how often to connect to the stored remote host. If set to 1, the module will only make one attempt to auto connect upon power up. If set to 2 or greater auto connect will re-open the connection after the connection is closed. Default=0 disables.

For auto connect timer to work, the remote host's IP address and port number needs to be configured in the WiFly GSX module.

Closing the TCP connection

In TCP-Client AND TCP-Server mode (default mode), there is also a disconnect timer. This timer can be used to automatically close a TCP connection after a specified number of seconds with no transmit or receive data.

set comm idle <secs>

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

The default value of the comm idle timer is 0, never disconnect on idle.

10.2. Wake on Sensor Input

SENSE 0 to 3 inputs wake the module from sleep. These pins have a small current source that is activated in sleep mode. This source is approximately 100nA, and will cause the input to float up to about 1.2VDC. If SENSE1 for example, is enabled, pulling the SENSE1 pin to GROUND will wake the device.

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

WARNING: Under no conditions should the voltage on any sensor input exceed 1.2VDC. Permanent damage to the module will result.

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

An open drain FET is a good device to tie to the SENSE pin. The threshold is about 500mV. Additional pullup to 1.2VDC may be used if the circuit has an impedance (due to leakage current) of less than 5Mohms (500mv / 100nA). SENSE 0 to 3 pins that are not used should be left unconnected.

10.3. Wake on UART

When the module is in sleep mode, the UART itself is disabled. However, wake on UART can be accomplished by connecting the SENSE pins to the RX data or CTS pin. (Using the appropriate divider resistors mentioned above)

The SuRF board (RN-134) has a built in resistor divider connecting SENSE 0 and SENSE 1 to RXD and CTS respectively. This allows wake on RX and CTS using a 3.3V signal.

NOTE: Do not apply 3.3V directly to SENSE 0 and SENSE 1. Under no conditions should the voltage on any sensor input exceed 1.2VDC. Permanent damage to the module will result.

NOTE: On SuRF board **rev 2** the resistor pack connecting 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.

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

It should be noted that the first (or possibly multiple) byte sent into the module will likely be lost, so the designer should take care to send a preamble byte to wake up the module before sending valid data bytes. A better way to do this is to use the CTS input to wake the module, and wait until it is ready to accept data. To enable this, use **set sys trig 2.**

10.4. UART Receiver, RTS/CTS Hardware Flow Control

The UART receive buffer is approx. 1500 bytes, and at lower baudrates (less than 115K) the system can psend data over TCP/IP without the need for flow control.

Depending on the frequency and quantity of data begin sent, the comm parameters will 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 will want to set the flush timer to a large number to avoid fragmentation. For high throughput cases increase the UART baudrate, set the flush size to 1460 and flush timer to a large value so full IP packets are sent.

You can control the packet forwarding 3 ways:

set comm match <value> sets the value of the packet terminator. Each time the match character is seen an IP packet will be sent. "set comm match 0xd" for example forwards once a 0xd hex character is seen.

set comm size <value> sets the flush size, the size is the number of bytes received before forwarding. Maximum value = 1460 bytes which is the size of a single Ethernet frame.

set comm time <value> sets the flush timer, this is used to make sure that any partial data sitting the RX buffer if no additional data is seen for “value” milliseconds. For example **set comm time 1000** would wait for 1 second after no data was sent.

When sending more than a few hundred thousand bytes in a single transaction you should enable hardware flow control. Your hardware will need to actively monitor CTS.

Flow control is not enabled by default. Flow control is set using with the following command.

set uart flow 1

It is possible to operate higher baudrates (greater than 115K) without flow control if packets are uniform and an application protocol is used to ensure that each 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 usually required whenever large, contiguous quantities of data are being written to the UART to guarantee no data is lost.

10.5. Setting GPIO direction, Alternate Functions and Disabling LEDs

The direction of the GPIO can be controlled with the GPIO mask using the **set sys mask <value>** command to set the GPIO pin direction. Value is entered as a hex number. If you need to set only one bit in the mask you need to read, mask and set the value. Otherwise you will over write any previous GPIO settings.

The hex value represents a bit mask that controls each pin where 1 = output and 0 = input. For example, **set sys mask 0x0** sets all pins to input.

To set only GPIO 6 and 7 for example, you would enter **set sys mask 0xc0**

The default mask for WiFly = 0x20f0, which has GPIO 13, 8, 7,6,5,4 as Outputs.

GPIO 0-3 are used internally on the module.
GPIO 4, 5, 6 are LEDs.
GPIO 9 is reserved as the ARM factory reset/adhoc mode, (read at power up) and otherwise general purpose input detect pin.

GPIO 10, 11 are the Uart RX, TX pins and TX does not need to be masked as an output.
GPIO12 is CTS (input) if used.
GPIO13 is RTS (output) if used.

The LEDs on the Surf Board (RN-134) are connected to GPIO 4, 5 and 6. To disable the LEDs, enter **set sys mask 0x20d0**

NOTE: The Yellow, Red and Green LEDs can be turned off. The Blue LED on the Surf board is the power LED and cannot be turned OFF.

The **get sys** command will show the setting of the GPIO mask.

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

The table below shows the usage of the GPIO pins with their default state and functionality.

Bit Position	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Signal Name	GPIO-13 UART RTS	GPIO-12 UART CTS	GPIO-11 UART-RX	GPIO-10 UART-TX	GPIO9	GPIO8	GPIO7	GPIO6	GPIO5	GPIO4	N/A	N/A	N/A	N/A
Default State	Output	Input	Input	Output	Input	Output	Output	Output	Output	Output	N/A	N/A	N/A	N/A
Default Function	Goes HIGH on POWERUP, LOW once system is READY. If HW Flow control enabled, toggles HIGH to indicate RX buffer full	Throttles transmitter if HW flow control is enabled. LOW enables transmitter, HIGH disable.	UART RX	UART TX	ADHOC MODE & FACTORY RESET	NOT USED	BLUE LED	RED LED	YELLOW LED	GREEN LED				

NOTE: The Blue LED is connected to GPIO7 on the Wi-Fi Serial Adapter (RN-370). The Blue LED is not connected to GPIO7 on the Surf Board (RN-134). It is not possible to power off the Blue LED on the Surf Board.

10.5.1. Setting the alternate GPIO functions

The defaults for GPIO 4 5, 6 is to control the LED functionality. This default functionality can be overridden to allow user programmable IO or alternate IO functionality by using the **set sys iofunc <value>** command. Value is entered as a hex number.

The hex value represents a bit mask that controls each bit in the <value> represents a particular GPIO pin. If a bit is 0, then that GPIO is driven/read by the firmware per the default function.

The IO function <value> is encoded as such:

Bit	IO	DIRECTION	Function
0	GPIO-4	output	Disable LED function so IO can be used as GPIO
1	GPIO-5	output	Disable LED function so IO can be used as GPIO
2	GPIO-6	output	Disable LED function so IO can be used as GPIO
3	Not Used		
4	GPIO-4	output	HIGH once associated/authenticated and have IP address.
5	GPIO-5	input	Set HIGH to trigger TCP connection, LOW to disconnect.
6	GPIO-6	output	HIGH when connected over TCP, LOW when disconnected.

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

If the LEDs are disabled using bits 0, 1, 2 above, you can then use the **show i** command to read these GPIO.

<2.21> **show i** will return
Port=30

For example, to use the alternate functions of the LEDs, the sequence of commands would be:

```
set sys iofunc 0x70          // enable alternate function for GPIO 6, 5 and 4
save                         // store configuration
reboot                       // reboot the module
```

NOTE: Currently, the alternative GPIO functions are not available in adhoc mode.

10.5.2. Controlling connections with GPIO

In embedded applications it is useful to monitor and control the status of the TCP/IP connection. This can be done by using the alternate function for GPIO-5 and GPIO-6.

With the alternate function for these GPIO set, the module will connect to the stored remote host IP address and port when GPIO-5 is driven high and disconnect when driven low.

The TCP/IP connection status can be monitored by reading GPIO-6, high = connected, low = not connected.

Here is how to set the WiFly module to connect using GPIO-5 and GPIO-6

```
<2.20> set ip host <addr>      // set the IP address of the remote host
<2.20> set ip remote <port>    // set the IP port of the remote host
<2.20> set sys iofunc 0x60     // set alternate function for GPIO-5 and GPIO-6
<2. 20> save                   // save settings in config file
<2. 20> reboot                 // reboot the module
```

On the remote host run your application or other software that opens and listens on the <port>.

Connect GPIO-5 to your embedded processor or other control signal. When GPIO-5 is driven high a connection will be attempted. When drive low the connection will be closed.

NOTE: DO not to drive the GPIO with more than 3.3 VDC or permanent damage to the module will occur.

If the connection to the remote host is successful GPIO-6 will go high. If the COMM OPEN and REMOTE strings are set you should see the *OPEN* messages on the UART and the *HELLO* at the remote host.

10.6. Setting Debug Print levels

There are a number of print functions that can be enabled to assist in debugging the operation and status of the module. The following command is used to control these printouts.

set sys printlvl <value> sets additional print functions. Value is a bit-mapped register that controls which printout messages are sent to the UART.

Print level	Description
1	All status messages
2	only critical NETWORK AP connection level status is output, "Associated!" Or "Disconnect from <ssid>"
4	DHCP and IP address status information Once the configuration has been checked; this can then be turned off so that these messages do not interfere with the data.

10.7. Using the Real Time Clock Function

The real time clock in the module keeps track of the number of seconds since the module was powered on and the actual time when synchronized with the sNTP time server. By default the module keeps track of up time but does not synchronize with the time server since this requires being associated with a network that can access the sNTP server.

The default sNTP server is at
ADDR=129.6.15.28:123
ZONE=7 (GMT -7)

Use the **show time** command to see the current time and uptime
<2.20> **show t**
Time=08:43:10
UpTime=10 s

Time can be set by using the **time** command
<2. 20> **show t**
Time NOT SET
UpTime=8 s

<2. 20> **time**

<2. 20> **show t**
Time=08:51:31
UpTime=15 s

NOTE: the WiFly module must by successfully associated with a network for the module to contact the sNTP server.

Alternatively, the module can be configured to get the time whenever it powers up by setting the time enable to 1. Any value greater than 1 gets time continuously every <value> minutes.

To configure the Wifly module to get time upon power up

<2. 20> **set time enable 1**
AOK
<2. 20> **get time**
ENA=1
ADDR=129.6.15.28:123
ZONE=7

To view a complete listing of the time variable use the command

<2. 20> **show t t**
Time=09:02:10
UpTime=653 s
Powerup=1792 s
RTC=7753271426558 ms
timera=66885

10.8. Time Stamping Packets

This feature can be used to automatically append 8 bytes to a TCP or UDP packet.

set ip flags 0x87 (enables timestamp and keeps other default settings)

TIME STAMP (MSB to LSB)

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 TCP checksum so it will pass thru 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 is also counting while in sleep mode.

11. Sending data using UDP

11.1. Overview

UDP is a connectionless protocol. There is no initial handshaking between the hosts to set up the UDP connection. There are no acknowledgements sent by the receiver for UDP packets that it receives. This makes UDP an unreliable protocol, as there is no guarantee that the data will be correctly delivered. However, due to its connectionless nature, UDP is suited for applications that cannot tolerate too much latency but can tolerate some errors in data. Transmission of video would be a good example of UDP application.

To use UDP on the WiFly-GSX module, you will need to enable the UDP protocol using the command "set ip proto 1". You will also need to specify the remote host IP address and the local and remote port number that you will use for UDP communications.

The commands to enable UDP data transfer are:

Associate to a network:

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

Set up the protocol and port number

```
set ip proto 1                  // enable UDP as the protocol
set ip host <ip address>        // set the IP address of remote host
set ip remote <port>            // set the remote port number on which the host is listening
set ip local <port>             // set the port number on which the WiFly module will listen
save                            // saves the settings in config file
reboot                          // reboots the module so that the above settings take effect
```

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

Since UDP is a connectionless protocol, data will start flowing as soon as the module is rebooted. Unlike TCP, it is not required to do an "OPEN" for the connection to be established. The WiFly-GSX module acts

like a data-pipe, so the UART data will be sent over the Wi-Fi link via the UDP protocol (in this case) and the data coming over the Wi-Fi link (via UDP protocol in this case) will be sent to the UART.

11.2. UDP Auto Pairing

UDP auto pairing feature temporarily stores the Host IP address of the first remote device that send a UDP packet into the module. This host IP address is stored in the RAM which will not survive a sleep or power cycle.

This feature allows the WiFly module to echo back to any client that sends a UDP packet. To use this feature, the host IP addresses and set the ip flags.

set ip host 0.0.0.0
set ip flags 0x80

11.3. 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 sent, (any UDP packet coming back in). If the response packet is not received by approximately 250 ms, the same UDP packet is sent out.

This continues until either

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

To enable this feature, use **set ip flags <value>**

11.4. Using the UDP Broadcast function

The WiFly module can be setup to automatically generate UDP broadcast packets. This is useful for a number of reasons:

- Some Access Points will disconnect devices that sit idle and don't send any packets after a time. Using the UDP broadcast informs the AP that WiFly is alive and wants to stay associated.

- This feature can be used by application programs to auto-discover and auto configure the WiFly 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 IP address and port number of the WiFly are both part of the packet, and thus the WiFly can be connected to and configured remotely with this information.

- The MAC address of the associated AP, channel, and RSSI value are available in this packet, thus enabling a simple location and tracking based function.

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

The format of the packet is: 110 bytes of data:

AP MAC address	Chan	RSSI	Local TCP port	Real Time Clock	Battery Voltage	GPIO pins	time of day	Version and datecode	User DEVICEID	Boot time	SENSOR pins

Bytes	Size	
0-5	6	MAC adddress of AP that we are Associated with (for location)
6	1	Channel we are on.
7	1	RSSI
8	2	local TCP port# (for connecting into the Wifly device)
10	4	RTC value (MSB first to LSB last)
14	2	Battery Voltage on Pin 20 in millivolts (2755 for example)
16	2	value of the GPIO pins
18	13	ASCII time
32	26	Version string with date code
60	32	Programmable Device ID string (set option deviceid <string>)
92	2	Boot time in milliseconds.
94	16	Voltage readings of Sensors 0 thru 7 (enabled with "set opt format <mask>")

NOTE: To add sensor data to the UDP broadcast message, the sensors have to be enabled using the sensor mask. **set q sensor 0xff** enables all sensors.

12. Joining Networks and Making Connections

Configuring the module to make connections is a two set process. First you need to associate with a access point (AP) and second you need to open a connection.

To configure the module over the WiFi link is a chicken and egg problem. The module must be associated to a network to connect to it and program the network settings. This problem can be solved by configuring the module from the UART or over the air using adhoc mode.

If configuring the module using adhoc mode, see section 15. Once in adhoc mode open up a telnet window on IP address 169.254.1.1 port 2000

If configuring the module using the UART mode either using the RS232 or development board, open a terminal emulator on the COM port associated with that deveice. The default baud rate is 9600, 8 bits no parity.

12.1. Associate with a network access point

From within the terminal window, put the WiFly GSX module into command mode by typing $$$ in the terminal window. You should get CMD back confirming you are in command mode.

Type ***show net*** to display the current network settings.

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

Now finding all available networks with the ***scan*** command

```
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 the network you're connecting to is open, you can simply use the join command to associate with the access point. From the scan list above you can see that roving1 is an open network access point. Type ***join roving1*** to associate with an access point.

```
<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
```

You could also have specified the roving1 access point by using the command ***join # 1***

If the access point is security enabled you will need to set the pass phrase prior to issuing the ***join*** command. The RN-131G module will attempt to inquire and determine the security protocol of the access point so you do not have to set the authentication mode. To set the pass phrase for WPA use the command ***set wlan phrase <string>***. For WEP set the key using the ***set wlan key <num>*** command.

Once you have successfully associated to the network the access point SSID is stored. This along with the pass phrase can be saved to the config file so the module can associate with the network each time it is booted up.

12.2. Making Connections

To make a connection into the module simply open a IP socket and connect to the IP address of the module. Telnet is a simple way to test this connection. From in Telnet type open <addr> <port>. In the example above the telnet command you look like ***open 10.20.20.62 2000***. Once open you can type characters into the UART window and see them on the Telnet window or visa versa.

To make a connection from the module you will need IP address and port number of your server application. A simple program to test this functionality is a COM port redirector. 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 is available from Pira at http://www.pira.cz/eng/piracom.htm

After installing and starting this program, note the IP address of the machine it is running on. This can be found by running ipconfig in the Microsoft command window.

With the WiFly-GSX module in command mode, type ***open <addr> <port>***. The server will report the connection is open and you can type characters into the UART window and see them on the server window or vice versa.

12.3. Setting up Automatic Connections

Often, it is desired on power up (or wakeup) to automatically connect out to a remote server, send data, and then disconnect. This can be configured to happen automatically.

In the following example assume the network SSID and security have been set correctly and autojoin is set to 1. This will also work in adhoc mode(autojoin 4), however there will be delay in connecting to the adhoc network from the remote computer so set the sleep timer large enough to allow the network to get set up and the autoconn establish a TCP connection.

When the module wakes up or is powered on the autoconn timer will cause the module to attempt a connection to the stored remote IP address and port. While this connection is open the sleep timer will not decrement. While data is flowing the idle timer will not decrement. Once data stops for 5 seconds the connection will be closed. The sleep timer will the kick in and put the module in deep sleep. Finally the wake timer will start the whole cycle again one minute later.

set ip host X.X.X.X (set up the IP address of the remote machine)
set ip remote_port num (set up the IP port of the remote machine)
set sys autoconn 1 (automatically connect out after 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 (wakeup after 1 minute of sleeping)

UART data TRIGGER mode. (version 2.19) This mode will automatically make a TCP/HTTP connection upon incoming UART data.

set uart mode 2

12.4. Controlling Connections using PIO5 and PIO6

PIO5 can be used to control the TCP connection. Once configured with the set system IO command the module will attempt to make a connection to the stored IP and PORT address when set high and will disconnect when set low.

set sys io 0x20 (configures PIO5 to connect/disconnect)

You can monitor the connection status by reading PIO6. High indicates an open connection, low indicates no connection. Use the command set system IO to enable PIO6.

set sys io 0x40 (configures PIO6 to represent the connection status)

12.5. Using DNS settings

WiFly contains a built in DNS client. If the IP address of the host is not specified (i.e it is set to 0.0.0.0), the DNS protocol will be used. WiFly will automatically attempt to resolve the host address stored with the command:

set dns name <string> sets the name of the host for TCP/IP connections.

Once the address is resolved an automatic connection will be made.

To manually lookup the IP address of a host, use this command:

Lookup <string> string is the hostname.

12.6. Utilizing the Backup IP address/connect function

WiFly contains a feature for auto-retry and redundancy. If the first IP host address connection fails, the backup IP will be used (if set). If this fails (or is not set) then the first DNS name will be used. If this fails (or is not set) then the Backup DNS name will be used.

To set the backup IP address, use:

set ip backup <address>

To set the backup DNS name, use:

set dns backup <string>

13. Using HTML client feature

The WiFly GSX module has a built in HTML client. When enabled, the WiFly GSX module is capable of getting or posting data to a web server. Using the HTML client, it is now possible to post serial and/or sensor data to the host web server. This feature make is possible to provide Wi-Fi capabilities to applications such as GPS units, remote sensors, weather station, etc.

Example: User wants to retrieve data from web server with this format:

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

Settings:

set ip proto 18 //enable html client
set dns name www.webserver.com //name of your webserver
set ip address 0 // so WiFly will use DNS
set ip remote 80 // standard webserver port
set com remote 0 // turn off the REMOTE string so it does not interfere with the post

To make the connection the command would be:

open

or inline you can send **open www.webserver.com 80**

The user's microprocessor should write 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 so the web server knows the page is complete.

13.1. Built-in HTML Client Modes

WiFly can be setup to automatically post data to and get data from a web server without any external HOST CPU. The advanced web features are enabled using the "set option format <mask>" command. This is a bit mapped register. The functions of the bits are described in the table below:

set option format <value> Bitmapped value.

Bit of Format reg	Function
0	Automatically send HTML data header based on broadcast interval.
1	Send users BINARY data (converted to ASCII hex)
2	Sample the GPIO and AtoD pins and format to ASCII hex
3	Appends **&id= < value>**, where **value** is the device ID string set with **set opt device <string>**
4	Appends **&rtc=<time>**, where **time** is real time clock value in message as 32 bit HEX value in format aabbccddeeff

Example:

To automatically send HTML data header, the command is **set option format 1**
To append sensor data in ASCII hex format, the command is **set option format 7**

13.2. Automatically periodically connect to web server

WiFly module can be setup to automatically post data to a web server. The auto connect feature is enabled by the **set sys auto <seconds>** command. For example, the WiFly module can be configured to connect to the web server every 10 seconds by using the **set sys auto 10.**

The example below illustrates the commands to configure WiFly for connecting to the web server every 30 seconds.

set com remote GET$/ob.php?obvar=WEATHER // setup the string
set sys auto 30 // auto connect every 30 seconds.
set option format 1 // auto send the header once connection is open
set ip proto 18 // turn on HTTP mode=0x10 + TCP mode = 0x2

NOTE1: *when HTTP mode is set, the WiFly automatically appends the* \n\n *to the end of the packet.*

NOTE2: *if the html header contains spaces, the $ is required when entering the string. Space is the command delimiter. When WiFly command parser sees $ it will convert this to a SPACE character.*

13.3. Automatically connect to web server on uart data

WiFly GSX supports a mode in which it can connect to the web server when it receives uart data. In this mode, connection to the web server will be triggered on uart data.

Example:

```
set ip proto  18                          // turn on HTTP mode=0x10  + TCP mode = 0x2
set dns name  www.webserver.com           //name of your webserver
set ip address 0                          // so WiFly will use DNS
set ip remote 80                          // standard webserver port
set com remote  GET$/userprog.php?DATA=   // sample server application
set uart mode  2                          // automatically  connect using data TRIGGER mode
```

Then when the serial UART data comes in, WiFly auto connects to the web server, and will automatically send:

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

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

13.4. Posting binary data:

Web servers expect ASCII data, so if the User data is binary, WiFly can convert binary data to ASCII format before sending it to the web server.

Example: using the same settings as above but this time use the

```
set ip proto  18                          // turn on HTTP mode=0x10  + TCP mode = 0x2
set dns name  www.webserver.com           //name of your webserver
set ip address 0                          // so WiFly will use DNS
set ip remote 80                          // standard webserver port
set com remote  GET$/userprog.php?DATA=   // sample server application
set option format 2                       //Converts user binafy data in ASCII hex format
```

If incoming UART data = 6 bytes of binary data with hex values 0x01 0xAB 0x03 0xFF 0x05 0x06

WiFly will send this string to the webserver:

GET /userprog.php?DATA=01AB03FF0506 \n\n

13.5. Auto posting sensor data:

WiFly module can send the value of the GPIO and sensor pins:

The data will come as 18 bytes of ASCII HEX: <2 bytes GPIO><channel 0 thru 7 sensor data>

set ip proto 18 // turn on HTTP mode=0x10 + TCP mode = 0x2
set dns name www.webserver.com //name of your webserver
set ip address 0 // so WiFly will use DNS
set ip remote 80 // standard webserver port
set com remote GET$/userprog.php?DATA= // sample server application
set q sensor 0xff // sets WiFly to sample all 8 sensor channels
set sys auto 30 // automatically make the connection 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 will be

GET /userprog.php?DATA=0F300000111122223333444455556666777\n\n

In the above example, the data format 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

13.6. Examples using the HTML client

Example#1: Auto posting sensor data:

In this example, we will connect to the web server at www.rovingnetworks.com/mike.php?ID= and send data "ID=1234" every 60 seconds. We will also append the sensor data to the "ID=1234".

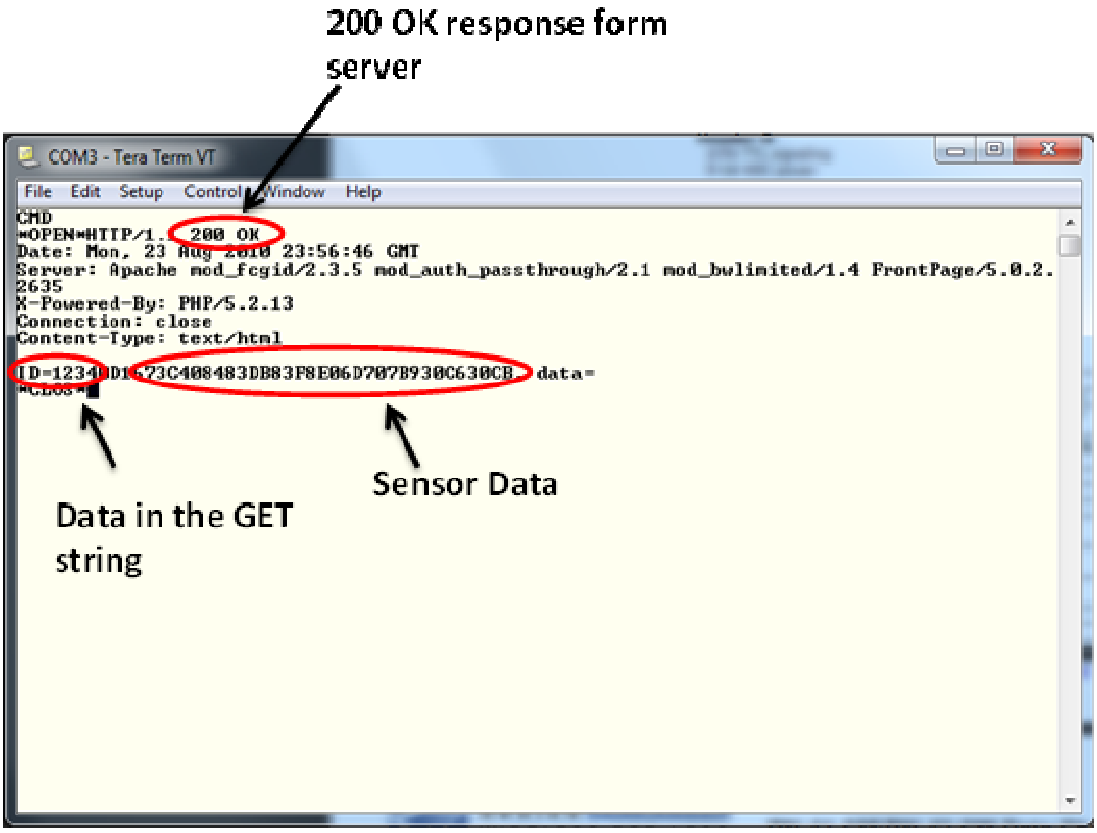
Set the network connections as described above. The other parameters that we need to set are described below.

```
Set dns name www.rovingnetworks.com      //set up the URL of the server
Set ip host 0                            //instructs RN-370 to use DNS address of host server
Set ip remote 80                         //standard web server port
Set ip proto 18                          //enable HTTP and TCP protocols
Set com remote GET$/mike.php?ID=1234     //set up the string
Set sys auto 10                          //auto connect every 10 seconds
Set option format 7                      //send the header and sampled binary data converted to ASCII
```

Set option sensor 0xFF	//sets sensor mask to sample all channels
Save	//save the configurations in config file
Reboot	//reboot so that the settings take effect

Result:

You will receive a 200 OK returned from the web server as seen in the screenshot below.



Example#2: Posting UART data to web server

The WiFly module is capable of auto posting serial UART data in ASCII or Binary format. In this example we will configure the WiFly module such that when the serial UART data comes in, the WiFly will connect and automatically send data to the web server in the following format:

GET /mike.php?ID=<user serial data> \n\n

The other parameters that need to be set are described below:

set dns name www.rovingnetworks.com	//set up the URL of the server
set ip host 0	//instructs RN-370 to use DNS address of host server

```
set ip remote 80                    //standard web server port
set ip proto 18                     //enable HTTP and TCP protocols
set com remote GET$/mike.php?ID=    //set up the string
set sys auto 10                     //auto connect every 10 seconds
set uart mode 2                     //automatically connect using data Trigger mode
save                                //save the configurations in config file
reboot                              //reboot so that the settings take effect
```

With the above settings enabled, the WiFly module will connect out 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" along with "set uart mode 2" will result in erroneous data being sent.

14. Firmware Upgrade over FTP

WiFly module has a file system for storing firmware and config files. Use the **ls** command to view files.. File size is displayed in sectors and the active boot image is identified in the final message.

FL#	SIZ	FLAGS	
11	18	3	WiFly_GSX-2.21
29	1	10	config

190 Free, Boot=11, Backup=0

Multiple firmware images and config files can be stored on the module file system.

NOTE: The Flash File system is used only to store firmware and configuration files. Currently the file system cannot be used to store data files

14.1. FTP Upload and Upgrade

WiFly contains a built in FTP client for getting files and updating the firmware. The client uses passive mode FTP, which allows operation thru firewalls and the Internet.

To update to the latest released firmware from Roving Networks the following setting are required:

FTP username = **roving**
FTP password = **Pass123**
FTP filename = **wifly-GSX.img**
FTP directory = **./public** (this parameter cannot be modified)

NOTE: To use FTP to upgrade the firmware, WiFly module has to be associated to an Access Point with internet connectivity.

To update the firmware, issue the following command:

ftp update <string> (string is an optional filename, use to bypass the default firmware filename)

The ftp update command will retrieve the file and switch the boot image to the new file.

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

The previous firmware will become the backup image. Here is an example of what you should see 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

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

Note the module must be rebooted or power cycled to use the new firmware. To boot a different firmware use the following command:

Boot image <num> sets the current boot image <num>

For example to boot the previous image from above use
<2.20> **boot image 11**
Set Boot Image 11, =OK

To upload your own firmware or config file to the module, change the stored FTP settings: See section 5.5 for more details on the FTP commands. To upload your file w following command:

ftp get <string> Retrieves remote file with name <string>

15. Adhoc Networking Mode

15.1. Infrastructure and adhoc comparison

There are two types of networks. The most common network is infrastructure in which an access point (AP) is the common point linking all Wi-Fi devices. The access point keeps track of who's on the local network and directs IP packets. In many cases the AP is also a router and will forward packets from the local network to other networks and the internet. It is also very common for the AP to be running a DHCP server which tracks and assigns IP addresses.

Adhoc is considered a point to point network in that each Wi-Fi device is linked directly to every other Wi-Fi device on the Adhoc network. There is no access point. All Wi-Fi devices on the adhoc 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 Auto IP, although one of the Wi-Fi devices can be configured as a DHCP server.

15.2. Configuring adhoc mode

The WiFly GSX module can be configured to setup an adhoc network. This mode is useful for point to point communications. The WiFly device is in Adhoc mode the device looks like access point for other Wi-Fi devices to join.

NOTE: Currently the WiFly only supports OPEN mode for creating adhoc networks.

Adhoc mode can be set via hardware or software commands.

To enable adhoc mode via hardware:

Set *PIO9* high (3.3V) at power up. On the RN-134 PIO9 is on J1 of the jumper block. When the module powers up in adhoc mode the WiFly module creates an adhoc network with the following

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 adhoc jumper in place the above settings override the current saved configuration settings.

To enable adhoc mode from software:

From command mode, the module is configured for adhoc mode using the join command. You will also need to set the ssid and channel.

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

Turn off DHCP and set the IP address and netmask so other devices know where to connect to the adhoc WiFly GSX. Since auto IP fixes the first two bytes of the IP address you want to use the netmask of 255.255.0.0 so that other device connecting to the module can be reached. Alternatively you can set the netmask to a smaller subnet if the other device's IP addresses are begin statically to the same subnet as the adhoc device

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

Be sure to save your configuration, then upon reboot the module will be in adhoc mode.

To associate with an adhoc network from another WiFly device:

set wlan ssid my_adhoc_network
reboot

or alternatively you can use the join command to associate with the adhoc network. Remember to disassociated using the **leave** command if you are previously associated to another network.

join my_adhoc_network

If you leave DHCP service enabled the WiFly device will get an IP address using auto IP when associating with the adhoc network. By definition auto IP fixes the first two bytes of subnet to 169.254.xxx.xxx. The WiFly device takes about two to three seconds to resolve the auto IP address.

Alternatively you can statically set the IP address by disabling the DHCP service and explicitly assigning the IP address.

set ip address 169.254.1.2
set ip dhcp 0

You can confirm the device has properly connected to the adhoc network using the ping command.

ping 169.254.1.1 10

To use associate with the WiFly adhoc network from another computer

Open the "Control Panel / Networking and Sharing / Networking and Sharing Center" dialog in Vista or "Control Panel / Network Connections" dialog in Windows XP. From here, view available networks and select the name of the adhoc network.

Note: Once associated with the adhoc network, Vista auto IP may take a couple minutes to allocate an IP address for your computer. To work around this you can assign a static IP address in the network settings / TCP/IP / Properties menu.

Once associated with the adhoc network you can open a connection or telnet window as you would with an enterprise connection.

NOTE: The module does not support adhoc and enterprise network modes simultaneously.

16. Analog Sensor Capability

The WiFly-GSX has 8 analog sensor inputs that can be driven between 0 to 1.2VDC. The analog inputs are sampled and the digital value read by using the **show q <channel>** command.

Warning: Over driving these inputs will cause permanent damage.

The hardware specifications of the analog input is:

Input voltage range: 0 - 1.2 V, however the A2D saturates at 400mV.
Resolution: 14 bits = 12uV
Sampling frequency: 35us
Accuracy: 5% un-calibrated

The accuracy of each analog sensor reading can be offset by up to 5% due to variation from chip to chip. To improve accuracy it is recommend using a precision reference voltage on one of the analog inputs to calculate the offset. The offset will be the same for all analog inputs.

For example,

- drive precision 200mV reference on analog input 4.
- read analog input 4 and compute the offset.

If you read 210mv you would know that the offset is +10mv.

When you read input 5 you would subtract 10mv from the result.

To read a sensor pin, send the following command:

show q <channel>

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 8xxxxx. The 8 in front is a start marker.

You can also sample multiple channels by using a bit mask:

show q 0x1<mask> where mask is a bit mask of the channels.

For example, to read channels 0, 1, and 7, send:

show q 0x183

The return values are the format: 8<chan0>, 8<chan1>, 8<chan7>\r\n

16.1. Automatic sampling of sensor pins:

The sensor pins can be automatically sampled and data forwarded in 2 modes:

1. the UDP broadcast packet will contain the value of the samples.
2. in HTTP mode, the pin sampled data can be forwarded to a remote server

to enable the above modes, use the

set q sensor <mask> command.

For example, to sample all sensors inputs, use the **set q sensor 0xff** command.

16.2. Using the Built In Sensor Power

WiFly modules contain an onboard Sensor power pin, which is controlled by the command below:

set q e <value>

The values used for setting the power are described in the table below:

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

ON BOARD TEMPERATURE OPTION (RN-121 TEMP)

show q t

The return values are the format: T=207\r\n this would be 20.7° C.

show q t 1 enables automatic sampling and output once per second.

show q t 0 turns off automatic sampling and output of temperature.

17. Default Configuration Settings

ADHOC PARAMETERS

Beacon	100 (milliseconds)
Probe	60 (seconds to look for beacons before declaring adhoc is lost)

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 byte	0
Flush timer	10 (milliseconds)
Idle timer	0
Cmd char	$

DNS PARAMETERS

IP address	0.0.0.0
Name	server1
Backup	backup2

FTP PARAMETERS

Server address	208.109.78.34 (roving default update server) (port at 21)
File	Wifly-GSX.img
User	roving
Password	Pass123

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)

OPTIONAL PARAMETERS

Device ID	WiFly-GSX

Join timer/WPA timer	1000
Replacement char	$ (0x24)
Format	0x00
Password	“” (no password enforced)
Sensor	0x00

SYSTEM PARAMETERS

Sleep timer	0
Wake timer	0
Trigger	1 (SENS0 pin wakes up the device)
Auto connect	0
IOfunc	0 (no alternate functions)
IOmask	0x21F0
Print level	1 (prints enabled)

TIME SERVER PARAMETERS

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

UART PARAMETERS

Baudrate	9600
parity	n (none, this is the only option available)
flow	0 (disabled)
Mode	0

WLAN PARAMETERS

SSID	roving1
Channel	0 (automatic scan)
External antenna	0 (off - use on-board chip antenna)
Join mode	1 (automatically scan and join based on ssid)
Authentication mode	OPEN
Mask	0x1FFF (all channels)
Rate	12 (24Mbit)
Passphrase	rubygirl
Key number	1
Key	0 0 0 0 0 0 0 0 0 0 0 0 0

17.1. Restoring Default configuration settings:

From command interface use the **factory RESET** command to restore the defaults. This command automatically loads default settings, and executes a “save” command.

From hardware, setting PIO9 high on power up arms the factory reset functional and toggling PIO9 five (5) times there after causes the configuration setting to restored to the factory reset.

PIO9 is sampled at about 1 Hz, so if using a CPU to generate the signal, make sure that PIO9 transitions (H to L or L to H) are at least 1 second long.

User file option

As of version 2.10 you can now specify a USER configuration as the factory reset settings. Prior to this release only the hardcoded factory defaults would be restored. If there is a config file named "user", it is read in as the factory defaults instead of using the factory hardcoded defaults. If no "user" config file is present, the hardcoded factory defaults are used.

The "user" config file is created using the "**save user**" command which saves the current configuration settings into the “user” file.

Even if there is a “user” config file arming and toggling PIO9 7 times will override the “user” settings and restore the wifly module to the factory hardcoded defaults. This is a bypass mechanism in case a bad configuration is saved into the “user” file.

Note: The module should be rebooted, or reset with reset line for the new settings to take effect.

18. Boot-up Timing Values

Function	Description	Time (in ms)
Power up	Powerup Time from Reset HIGH or power good to boot code loaded.	70
Initialization	Initialize ECOS	50
Ready	Load configuration and Initialize application	30
	Total time to READY	150
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 2 (highly dependent on Access Point response)	50 - 250
Aquire IP	DHCP obtain IP address (highly dependent on DHCP server response time)	30-???

19. Supported Access Points

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

We also currently do not support WPA2-Enterprise (radius server authentication, EAP-TLS)

The WiFly GSX should work with any standard Access Point. We have tested the WiFly-GSX module with the following access points:

- Cisco Aeronet series
- Linksys (both standard and openWRT Linux)
- Netgear WGR614 v8
- Netgear WGN54
- DLINK dir-615
- Airlink 101
- Apple Airport express
- Buffalo networks
- ADHOC MODE (Apple iPhone, Microsoft windows PC with XP, Vista , Ubuntu Linux)

20. Release Notes

20.1. Known problems

- **The UART does not support odd or even parity, only no parity is supported.**
- **Flow control: RTS may fail to de-assert quickly enough for some high speed CPUs to correctly stop sending data bytes. For high speed transfers at baudrates > 460800, it is best to limit RX data to the maximum Ethernet frame (1460 bytes) and have a protocol to acknowledge data is received by the remote host.**

20.2. Current Firmware features and fixes

As of 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.

As of 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. Refer section 10.6 for more deails

- Ability to add prefix to HTML client post, specifically the ability to append **&id=** and **&rtc=** in the HTML message. Please refer section **Error! Reference source not found.** for more details.

As of Version 2.19 3/05/2009

Fixes

- Improved performance of the UART receiver. UART is now reliable at up to 460Kpbs with RTS flow control.

Features

- Created UART data trigger mode, which will automatically make a TCP/HTTP connection based on received UART data. **set uart mode 2** to enable this mode.
- Added timestamping option to both UDP and TCP packets. 8 byte RTC counter is appended.
- DHCP client now inserts the DEVICEID string into the HOST name when requesting a DHCP lease. This string is displayed by most routers and DHCP servers in their lease tables.
- **show n n** command returns the MAC Address of the Access Point currently associated.
- **get i a** command returns only the IP address of the WiFly.
- **show network** added a response "Boot=<time in ms>" which displays the total time in milliseconds that was required to be ready on the network (associate and get IP address). This time is also added to the UDP broadcast packet at byte location 92.
- Added a number of HTTP commands for posting data to a webserver see Section 12**.**

As of Version 2.15 10/15/2009

- Fixed a problem whereby the first UART RX character received on power up is received but does not sent until receipt of 2nd character.
- Fixed a problem with some APs that violate Wi-Fi specifications by not responding to WPA authentication within 250ms. The **set option jointimer xxxx** command, which specifies the

timeout in ms for a join now also applies to the WPA timeout. The default is now 1000ms or 1 second. Note: some APs require up to 1500ms to respond.

- When connected over TCP and the AP disappears or WiFly loses association the WiFly will now closes the connection. The *CLOS* response will now appear when the connection is terminated by the WIFly. NOTE: This may require the use of the **set comm idle xx** setting to monitor the TCP connection, and force a TCP disconnect when no data is flowing due to lost association.

Features

- **Link monitor** The command **set wlan linkmon x** is now used to monitor the state of the association to the AP. The AP is scanned once per second, and if x consecutive scans fail, the WiFly declares "AP is lost" sets the interface to down state, and enters the association process. Previously the WiFly would not detect that the AP association was lost until the AP became available again, or the WiFly was power cycled or rebooted.

- **ADHOC mode** The command **set adhoc probe x** is now used to set a threshold for the number of consecutive missed probe responses allowed before declaring "ADHOC is Lost" and setting the network interface to be down. Default is 5 probes. A setting of **set adhoc probe 0** will disable this function. Some Adhoc stations do not reliably respond to probes and so this value higher avoids intermittent loss of connectivity.

- **DHCP cache** The **set ip dhcp 3** command is now used to enable DHCP address caching. Once caching is turned on, the initial DHCP settings are stored in NVRAM. This is most useful in battery systems, when using the sleep mode. Upon waking from sleep, as long as the DHCP lease time is still valid and the WiFly is associated to the same AP, DHCP caching does not survive a power cycle or usage of the hardware reset pin.

- **ARP table cache** The **set ip flags 0x20** command is now used to enable ARP table caching. Once caching is turned on, any ARP table settings are backed up to NVRAM before sleep. Upon waking from sleep, the ARP cache is loaded. ARP table caching does not survive a power cycle or usage of the hardware reset pin.

- **DNS host address cache.** The **set ip flags 0x10** command enables DHCP address caching. Once caching is turned on, the initial DHCP settings are stored in NVRAM. This is most useful in battery systems, when using the sleep mode. Upon waking from sleep, as long as the DHCP lease time is valid and the WiFly is associated to the same AP, DNS caching does not survive a power cycle or usage of the hardware reset pin.

- **UART break detect enables sleep.** The command **set uart mode 8** enables break detection on the UART RX pin. Once Break is detected (a consistent low value on RX pin) ,WiFly waits for the UART RX pin to return to a high value before going to sleep.

- **UART NOECHO mode.** The command **set uart mode 1** is now used to disable echoing of RX chars while in command mode. This is useful when embedded controllers are used to send

commands to the module. NOTE: For consistency, the command prompt response **<2.xx>** now also contains **\r\n** appended string when in this mode.

As of Version 2.12 9/17/2009

- Fixed problem with some newer 802.11n - association attempts cause module to crash/reboot. (Such as Linksys WRT160NL)
- Fixed problem with send on match character i.e. set comm match char. Match char is now operational.
- During an open TCP session, a second incoming connection would be accepted. Second connection is now accepted but then immediately closed.
- Hardware flow control is now supported. To enable, use the **"set uart flow 1"** command.
- DHCP renew and rebind is fully supported. Previously, DHCP renew/rebind would update IP settings, and if a TCP session was active it would enter a hung state. TCP connections now survive a DHCP renew/rebind.

Features

- **TCP connection password**. This optional pass word is enabled with the command "set opt pass <string>", incoming connections will be challenged and the stored password must be matched or the connection will be closed.

- **UART instant baudrate** The **set uart instant <rate>** command immediately changes the baudrate. This is useful when testing baudrate settings, or switching baudrate "on the fly" remotely while connected over TCP.

- **Analog interface commands** The "show q" command will now enable and show the digital value of the analog interface pins. See section 16

As of Version 2.11 9/8/2009 – Limited release (please update to 2.12)

As of Version 2.10 8/14/2009

- Added a 250ms guard band in parsing of $$$. The module now looks for three $$$, and only three $$$ within a 250ms period with no additional characters following for 250 ms. **Do not send \cr or \lf after the $$$.**
- Fixed problem with UART dropping data. In cases with large data transfers (>100KB) the UART would become over whelmed and drop data.
- We no longer pass serial data received into the UART back over telnet when in remote command mode

Features

User specified default configuration - You can now specified a USER configuration as the factory reset settings. The function of PIO9 has been changed slightly. See section 17.1

Configurable Device ID – There is now an additional user programmable device ID that can be used for storing serial numbers, product name, device type or other information. The device ID is part of the broadcast "hello" UDP message that the module sends out to identify itself. Use the command show deviceid to display the current setting. For more information on using this command see the "set optional" section command

UDP broadcast packet – By default the WiFly module now sends out a UDP broadcast to 255.255.255.255 on port 55555 at a programmable interval. The broadcast address, port and interval are set using the set broadcast commands. See section 11

Known Issues

WiFly Module has trouble associating with some 802.11.n access points. The module will crash and reboot repeatedly. We have seen this behavior with Linksys and Dlink router/access points. If you disable the .n capability on the router the module will associated correctly.

Flow control is not functional.

Current Firmware Version 2.09 7/10/2009

- Sleep mode was drawing 70uA instead of the expected 4uA due to an oscillator that was not disabled before going to sleep. Refer to the RN-131G datasheet for the proper low-power hardware configuration.

- Fixed closing of TCP port on TCP RESET. Previously the module was not handling remote TCP reset correctly and would disconnect which resulted in a printout of ERR= -5, TCP port was not closed properly.

- Fixed clearing and setting of strings in several set commands. In these cases the strings could be erased, but not reset.
    - set comm remote
    - set comm open
    - set comm close
    - set dns name
    - set dns backup

- Removed extra character in UART output. Previously the module would insert an extra "\r" character when '\n' appears in data stream.

- Added the **get everything** command to dump out all configuration settings

- Fixed the alternate I/O functions to allow connection based on PIO5. The manual has been updated to include a much better description of this functionality. See section 10.5

As of firmware version 2.08 6/08/2009

- Connecting out an IP address does not use the DNS and backup DNS if the connection to the primary IP address fails. Connecting using DNS if the IP address if 0.
- UART hardware flow control not yet functional.
- TCP_NODELAY added as default. This improves performance as the stack no longer waits for each TCP packet to be ack'ed, (since many Microsoft systems only ack every OTHER packet).

- Set ip proto is now a bitmask. It is possible to have both UDP and TCP bits set. If TCP enabled, UART RX data will be forwarded via TCP if a connection exists. Otherwise, data will forward over UDP (if UDP bit is set).

As of firmware version 2.07 6/04/2009

Command changes

- set wlan antenna < 0 or 1 >command has been changed to set wlan extant <0 or 1 >.
- set wlan auth <value> command has been added
- **set wlan hide** will hide the WEP key or WPA passkey. To unhide, you set key or passphrase again.
- **set ip proto 8** TCP client mode, (no listen server) only outbound connections can be made.

- Bug fixes
- Adhoc mode client associates properly
- You can now enter the WPA passkey after setting the SSID, previously the pass key had to be entered first for the security hash to be correctly created.
- Auto join now stops after 3 retries.

Features

- **show net** now displays the WiFi TX rate, and correctly displays authenticated state and shows authentication mode that was used.
- **ping h** will ping the stored host address. If no host address stored, will attempt to use the DNS hostname.
- **ping i** command added to ping a known Internet server (www.neelum.com) by first resolving the address, proving that DNS is working and then pinging the server. This proves the device has internet connectivity.
- UDP secure mode will only forward packets to the UART that match from the host address. TCP secure mode will only allow connection from and IP that matches host address.

As of firmware version 2.06

- Web server interface is not available – Configuration over telnet and the UART
- UART flow control is not functional – The module may drop data at high data rates
- Sensor pins for reading analog signals are not supported
- Wake on UART RXD or CTS is not working on current revision REV2 of the SurfBoard.

- The fast- auto sleep timer for UDP mode is not implemented.

Fixes since firmware version 2.05

- Configuration over Telnet not functional
- Error checking the correct number of parameters