WISNAP WIFI/SERIAL ADAPTER

802.11 B/G Wireless LAN Module

User Manual and Command Reference

Version 2.15
Apr 6th, 2010

1 Overview

The “WiSnap” radio module is a complete stand alone embedded wireless LAN access device. The device has an on-board TCP/IP stack and RS-232 interface for transferring data to remote applications, such as an iPhone app, data logger, or PC control console.

- Fully Qualified and Wifi 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 Ad-hoc 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 2.7 Mbps for UART
- Memory 128 KB RAM,2MB ROM, 2 KB battery-backed memory, 8 Mbit Flash.
- Intelligent, built-in power management with programmable wakeup
- Can be powered from regulated 3.3-3.7V 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 WiFi authentication WEP-128, WPA-PSK (TKIP), WPA2-PSK (AES).
- Built in networking applications DHCP client, UDP, DNS client , ARP, ICMP ping, FTP, HTTP
- 802.11 power save and roaming functions

One of the main applications for this device is the iPhone, since it requires buying additional authorization hardware to use Bluetooth SPP, the WiSnap in ad-hoc mode is a simple and cost effective way to connect to iPhone apps. The WiSnap Serial Adapter is more than a cable replacement solution. By allowing multiple TCP/IP sockets, applications can control and monitor hundreds of WiFi Serial adapters remotely distributed across a building LAN or campus WAN.

2 Hardware Interface

2.1 Power

There are two options for powering the WISNAP 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. 3.3V-REG-OUT (Pin 17) leave floating.

BATTERY: Apply battery = 2.0 to 3.3VDC to VBATT (pin 20). V3.3IN pin 21 = floating. Tie pin 17 to pin 18. (This enables the on board battery boost 3.3V switcher).

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

Warning: Do NOT exceed the voltage ratings on the 3.3V pins, this may result in damage to the module.

Notes:

#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.

When using a 5.0 VDC Input, PIO's and UART input pins require a resistor divider. A suggestion is 10K in series with 20k to ground. PIO's are 0-3.3 VDC not 5.0 VDC tolerant.

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.

State	Red LED	Yellow LED	Green LED	Blue LED
ON solid	Not Associated		Connected over TCP	Full Charge
Fast blink		Rx/Tx data transfer	No IP address or Config Mode	
Slow blink	Associated, No Internet		IP address OK	Low Power
OFF	Associated, Internet OK			

3 Configuration

3.1 Entering Command Mode

You can administer the device with your favourite terminal emulator, such as Teraterm or PuTTY.

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. Defaults are: 9600 baudrate, 8 bits, No Parity, 1 stop bit, and hardware flow control disabled.

Typing **"$$$"** on in the terminal emulator will enter command mode. You should see "**CMD**" returned to you. This will verify that your cable and communication 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.

Remote configuration using AD-HOC mode

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

To enable ad-hoc mode, enable the J1 switch on the jumper block. When the module powers up with this jumper set on, the WiSnap module creates an ad-hoc network with the following settings:

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

With the ad-hoc jumper in place, the above settings override the current saved configuration settings.

From your computer, connect to the WiSnap-XX network. This is an open network which does not require a pass phrase or pass key.
NOTE: currently the WiSnap only supports OPEN mode for creating ad-hoc 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 a command window. If connected, this command will show you the IP address and netmask for your computer.

Your IP address must be on the 169.254.1.X network otherwise the WiSnap module will not be accessible.

NOTE: If your machine has both a wireless and wired network interface, you may need to disable the wired LAN interface hardware before connecting to the ad-hoc network. If the wired LAN is enabled, the computer may assign an IP address that is not on the same subnet as the WiSnap module.

Once connected and you have a valid IP address, telnet into the WiSnap module on port 2000 with this command in Windows:

```
telnet 169.254.1.1 2000
```

You should see the response "*HELLO*"

You can now enter command mode and configure the module.

3.2 Common Configurations

Two common modes of operation for the WiSnap module are A) initiating a connection to a server and B) listening for a remote host connection. This section will go through the configuration for each setup. The setups are shown using infrastructure network. i.e. with an access point, however the same can be done with ad-hoc networking.

Initiating a connection from the WiSnap

Step 1: Set up the WLAN properties so the device will connect to the network automatically upon power up. In this example we want to connect to the wireless network my_network.

```
set wlan join 1                      // Auto join upon power up
set wlan chan 0                      // Scan all channels
set wlan ssid my_network             // Network name
set wlan phrase my_secret_code       // Pass phrase
```

The **join 1** setting ensures that when the module wakes up, it tries to join the access point that matches the stored SSID, passkey and channel. Channel =0 (the default) will force auto-scanning. Setting the channel will reduce the time it takes the WiSnap to find and associate.

Step 2: Set up the IP address and port number of the remote server, so the WiSnap can connect when it wakes up.

```
set ip host 10.20.20.75         // Set the host IP address
set ip remote 3000              // Set the remote port
set sys autoconn 2              // Try to connect to the host every 2 seconds
save                            // Save configuration>
```

Note: If autoconn = 1, the WiSnap will only make one attempt to auto connect.

Step 3: Set the wake up and sleep conditions. By default the adapter will wake whenever there is data written to the serial interface. You can also configure the device to wake up on CTS, on a PIO or timer. See the command reference for details. We are going to set this up to wake on a timer then sleep after 2 minutes if there is no connection or if connected and no data has been transferred for 30 seconds.

```
set sys sleep 120          // sleep after 2 minutes if no connection
set sys trigger 2          // wake on CTS
set conn idle 30           // disconnect after 30 seconds of no data
save                       // save all the settings to the config file
reboot                     // use the new settings
```

This setup can be tested using TCP server application that opens a socket on port 3000. Port Peeker is a free application that you can download off the web.
It is available at http://www.linklogger.com/portpeeker.htm

Waiting for the remote host to connect to the serial adapter (listen mode)

In this example we are using a static IP so that the remote host knows where the WiSnap Serial adapter is on the network. Alternatively you can write your application software to listen for the broadcast UDP packet (automatically sent by WiSnap by default) to identify the WiSnap Serial adapter and get the IP address and TCP port number that the WiSnap is listening on.

Step 1: Set up the wlan properties so the device will connect to the network automatically upon power up. In this example we want to connect to the wireless network my_network.

```
set wlan join 1                    // Auto join upon power up
set wlan chan 1                    // only look on channel 1
set wlan ssid my_network           // Network name
set wlan phrase my_secret_code     // Pass phrase
```

Step 2: Configure the WiSnap static IP address so the remote application can connect, turn off DHCP and set the IP address and netmask.

```
set ip address 10.20.20.63       // Set the IP address
set ip port 5030                 // Set the local port to listen on
set ip netmask 255.255.255.0     // Set the IP netmask
set ip gateway  10.20.20.1       // Sets the network gateway
set ip dhcp 0                    // Turn off DHCP
```

Step 3: Set the wake up and sleep conditions. In this mode the sleep and wake timers are used to conserve battery. Since we don't know when the remote host will connect, the module should to occasionally wake up and listen for the remote host. The trade off with these timers is that the longer you sleep, the better your battery life will be but the longer it will take the remote host to connect.

WARNING: Do not set the sleep timer below 5 seconds or it will be impossible to get into command mode to reprogram this mode without it going back to sleep!

```
set sys wake 20             // Wake after 20 seconds
set sys sleep 10            // Go to sleep after 10 seconds
save                  // Save configuration
reboot                // restart using the new configuration
```

At this point you could test this configuration using telnet on a computer sharing the same network to connect to the WiSnap module.

4 WiSnap 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" before the value: **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 – Changes settings immediately and 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: At power up, you will only associate, set the channel and get your IP address once.

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.

AD-HOC	**-** controls the ad-hoc 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 baud rate and parity
WLAN	**-** wireless interface settings, such as SSID, channel, and security options

5.1 AD-HOC Parameters

set ad-hoc beacon <ms> sets the ad-hoc beacon interval in milliseconds. Default is 100.

set ad-hoc probe <num> sets the ad-hoc probe retry count. Default is 5. This is the number of consecutive probe responses that can be lost before declaring "AD-HOC 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 10.4.
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.
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 intervals. 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 10.4.

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 WiSnap module. If DHCP is turned on, the IP address is assigned and overwritten during association with the access point.

Example: "set ip a 10.20.20.1"

set ip backup <addr> sets a secondary host IP address.

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 Ad-hoc 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 IP related advanced functions. Value is a bit mapped flag register. Default = 0x7.

Bit	Function
0	TCP stack copies RX buffer before sending
1	Bypass Nagle algorithm and use TCP_NODELAY

2	TCP application level single retry enabled
3	RETRY multi - retries 4 times
4	DNS host address caching enabled
5	ARP table caching enabled
6	Reserved
7-31	Reserved

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 WiSnap 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 WiSnap 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	Future Use

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 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 WiSnap 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 "WiSnap<DEVICEID>M1".

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 WiSnap module will close the connection. When the password is set the WiSnap 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.

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 RS-232 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 <value> 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

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 WiSnap module interrogates the Access Point and automatically selects the authentication mode.

The current release of WiSnap 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	Ad-hoc, Join any Ad-hoc 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 Ad-hoc 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 Ad-hoc device can act as DHCP server) This policy is often used instead of the hardware jumper to creat a custom Ad-hoc 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 WiSnap 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, WiSnap will scan once per second for the AP it is associated with. The value is the threshold of failed scans before the WiSnap declares "AP is Lost", de-authenticates. The WiSnap 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 WiSnap 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 ad-hoc display all ad-hoc 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 display IP address and port number settings**.**

get mac display the device MAC address**.**

get optional display the optional settings like device ID

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 Displays current network status, association, authentication, etc.

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**. After this command the new settings must be save to the config file using the ***save*** command and the module rebooted for them 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 from currently associated Access Point.

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>
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 Ad-hoc mode (version 2.11).

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

10.1 System Timers and Auto Connect Timers

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 timer is reset when characters are received on the UART.

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

UDP sleep, and Connection timers

There is another timer than can be used to put the device to sleep.

In UDP protocol mode, 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.

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

the timer is decremented every xx milliseconds, based on the value of the comm flushtimer. 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.

In TCP-Client mode, the auto-conn timer is used as a connect out timer. If set, the device will automatically attempt a connection when the timer expires.

set sys autoconn <secs>

In TCP-Client AND TCP-Server mode, there is also a disconnect timer.

set comm idle <secs> sets the idle disconnect timer. This causes a disconnect if no transmit or receive data is seen.

10.2 Wake on Sensor Input

SENSE 0 to 3 inputs are available to wake the module from sleep.
SENSE 0 to 3 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 these inputs to wake the module, use the command **set sys trigger <value>.** The value is a bit-mapped setting. To wake on sensor pin 2, use **set sys trig 4**, for example. Setting the value to 0 disables all sensors pins.

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.

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

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 SENSE1 pin to the RX data 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 to allow waking up the module.

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 Optimizing UART RX and Hardware Flow Control

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

Depending on the frequency and quantity of data begin sent comm parameters will optimize Wi-Fi performance by specifing 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 maximum size of a single Ethernet frame.

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 WiSnap = 0x21f0, 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/ad-hoc mode, (read at powerup) 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 **get sys** command will show the setting of the GPIO mask.

```
<2.09> get sys
SleepTmr=0
WakeTmr=0
Trigger=0x1
Autoconn=0
IoFunc=0x0
IoMask=0x21f0
PrintLvl=0x1
```

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	GPIO 9	GPIO 8	GPIO 7	GPIO 6	GPIO 5	GPIO 4	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	AD-HOC MODE & FACTORY RESET	NOT USED	BLUE LED	RED LED	YELLOW LED	GREEN LED				

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 programable 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.

```
<2.09> show i     will return
Port=30
```

You can also use the **set sys output <value> <mask>** to drive GPIO output pins to HIGH or LOW. (mask is optional, default sets all the pins).

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 WiSnap module to connect using GPIO-5 and GPIO-6

```
<2.09> set ip host <addr>     // set the IP address of the remote host
<2.09> set ip remote <port> // set the IP port of the remote host
<2.09> set sys iofunc 0x60    // set alternate function for GPIO-5 and GPIO-6
<2.09> save
<2.09> reboot   // the module must be rebooted for the alternate settings to take effect
```

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. **Be sure to not to drive the GPIO with more that 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. Bit-mapped value.

Bit	Function
1	Print start-up messages showing progress of association, dhcp, etc.

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

```
<2.09> show t
Time=08:43:10
UpTime=10 s
```

Time can be set by using the **time** command

```
<2.09> show t
Time NOT SET
UpTime=8 s

<2.09> time

<2.09> show t
Time=08:51:31
UpTime=15 s
```

NOTE: the WiSnap 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 WiSnap module to get time upon power up

```
<2.09> set time enable 1
AOK
<2.09> get time
ENA=1
ADDR=129.6.15.28:123
ZONE=7
```

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

```
<2.09> 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.

10.9 Using the UDP Broadcast function

The WiSnap 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 WiSnap is alive and wants to stay associated.
- This feature can be used by application programs to auto-discover and auto configure the WiSnap 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 WiSnap are both part of the packet, and thus the WiSnap 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 WiSnap 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			Local	Real	Battery		time of	Version and	User	Boot	SENSOR

address	Chan	RSSI	TCP port	Time Clock	Voltage	GPIO pins	day	datecode	DEVICEID	time	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 number (for connecting into the WiSnap device)
10	4	RTC value (MSB first to LSB last)
14	2	Battery Voltage on Pin 20 in millivolts (2755 for example)
16	2	Digital sensor option (Temperature and humidity)
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 through 7 (**set opt format <mask>**)

11 Joining Networks and Making Connections

Configuring the module to make connections is a two-step process. First you need to associate with a network access point 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 ad-hoc mode.

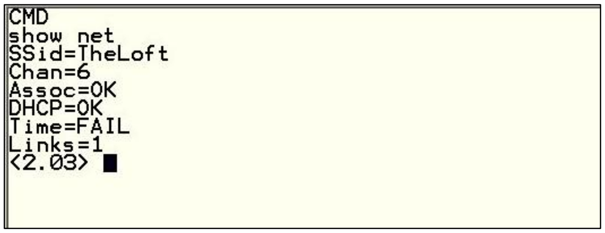
If configuring the module using ad-hoc mode, see section 14. Once in ad-hoc mode open up a telnet session on IP address 169.254.1.1 and port 2000.

If configuring the module using the UART mode either using the RS-232 or development board, open a terminal emulator on the COM port associated with that device. The default baud rate is 9600, 8 data bits, 1 stop bit, and no parity.

11.1 Associate with a network access point

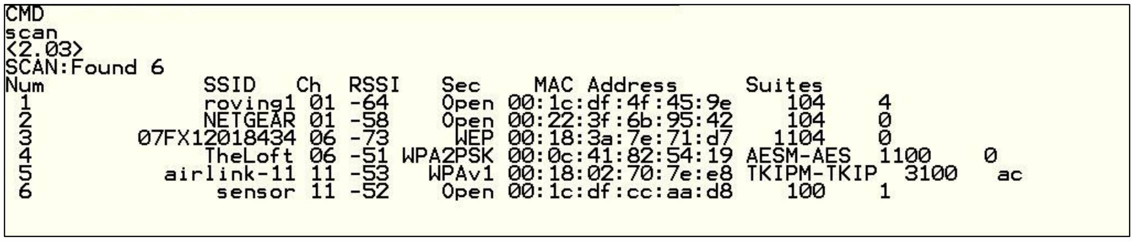
From within the terminal window, put the WiSnap module into command mode by typing $$$ in the terminal window. You should get CMD back confirming you are in command mode.

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



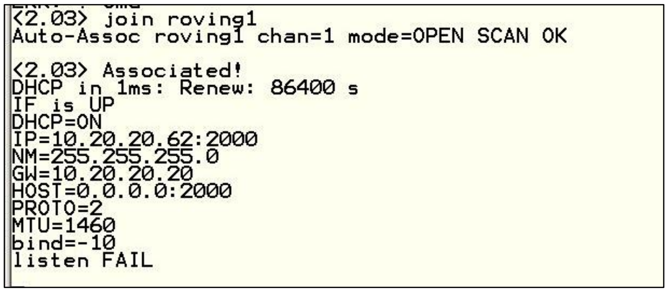
```
CMD
show net
SSid=TheLoft
Chan=6
Assoc=OK
DHCP=OK
Time=FAIL
Links=1
<2.03>
```

Now find 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 the 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 can also specify the an access point by using the number from the scan with 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 WiSnap 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 networks, use the command ***set wlan phrase <string>***. For WEP networks, 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.

11.2 Making Connections

To make a connection into the module simply open an 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 from connecting to an access point 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 WiSnap 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 visa versa.

11.3 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 ad-hoc mode(autojoin 4), however there will be delay in connecting to the ad-hoc 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 (new in firmware version 2.19) - This mode will automatically make a TCP/HTTP connection upon incoming UART data.

set uart mode 2

11.4 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)

11.5 Using DNS settings

WiSnap 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), DNS will be used.   The WiSnap 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.

11.6 Utilizing the Backup IP address/connect function

WiSnap 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>

12 Sending and Receiving Data via Web Server

The WiSnap can be setup to easily pull or post data to a web server.

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

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

Settings:

```
set dns name  www.webserver.com
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 to inline, you can send **open** www.webserver.com **80**

The user's microprocessor should then send this data:

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

Where the \n is the LINEFEED ASCII character decimal 10 or hex 0xA
Two linefeeds are required so the web server knows the page is complete.

12.1 Automatic-Advanced Web Server Modes

The WiSnap can be setup to automatically post and get with special HOST CPU support required!

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
3-7	Reserved for future use

12.1.1 Reading a page

The WiSnap can be setup to automatically post with no HOST CPU:

set com remote GET$/ob.php?obvar=WEATHER   // setup the string.

set sys auto 3   // auto connect every 3 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.*

12.1.2 Posting data to a page

WiFly can auto-post Serial UART data in ASCII or BINARY.

Example: data is an ASCII string...

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, the WiSnap will auto connect, and will automatically send:

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

12.1.3 Posting binary data

Web servers expect ASCII data, so if the data to be sent is binary, the WiSnap can convert it.

Example: using the same settings as above but with this change: **set option format 2**

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

12.1.4 Auto posting sensor data:

The WiSnap can send its own data which is comprised of the value on the GPIO pins plus the sampled data on the sensor pins:

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

set sys auto 3 // automatically make the connection

set option format 7 // send the header plus the sampled binary data converted to ASCII

The resulting string sent to the server would be:

```
GET /userprog.php?DATA=0F3011112222333344445555666677777\n\n
```

13 Firmware Upgrade over FTP

The WiSnap module has a file system for storing firmware, web pages 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-2.18
 29    1      10             config

190 Free, Boot=11, Backup=0
```

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

FTP Upload and Upgrade

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

To upload 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 can not be modified)

To use FTP to upgrade the firmware, enter the following command:

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

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

```
<2.10> ftp update
<2.10> 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-2.05
```

```
29      1       10                config
30      18      3                 WiFly_GSX-2.10
```

208 Free, Boot=30, Backup=11

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
<2.10> **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 use following command:

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

14 Ad-hoc Networking Mode

14.1 Infrastructure and ad-hoc 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 WiFi 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.

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

14.2 Configuring ad-hoc mode

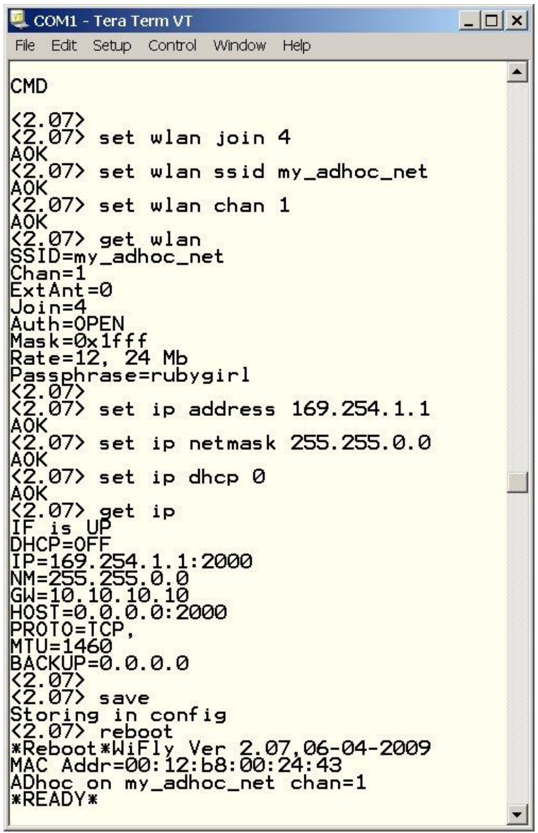
The WiSnap module can be configured to setup an ad-hoc network. This mode is useful for point to point communications. The WiSnap device is in Ad-hoc mode the device looks like access point for other WiFi devices to join.

Note: currently the WiSnap only supports OPEN mode for creating ad-hoc networks.

Ad-hoc mode can be set via hardware or software commands.

To enable ad-hoc 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



```
CMD

<2.07>
<2.07> set wlan join 4
AOK
<2.07> set wlan ssid my_adhoc_net
AOK
<2.07> set wlan chan 1
AOK
<2.07> get wlan
SSID=my_adhoc_net
Chan=1
ExtAnt=0
Join=4
Auth=OPEN
Mask=0x1fff
Rate=12, 24 Mb
Passphrase=rubygirl
<2.07>
<2.07> set ip address 169.254.1.1
AOK
<2.07> set ip netmask 255.255.0.0
AOK
<2.07> set ip dhcp 0
AOK
<2.07> get ip
IF is UP
DHCP=OFF
IP=169.254.1.1:2000
NM=255.255.0.0
GW=10.10.10.10
HOST=0.0.0.0:2000
PROTO=TCP,
MTU=1460
BACKUP=0.0.0.0
<2.07>
<2.07> save
Storing in config
<2.07> reboot
*Reboot*WiFly Ver 2.07,06-04-2009
MAC Addr=00:12:b8:00:24:43
ADhoc on my_adhoc_net chan=1
*READY*
```

up in ad-hoc mode the WiSnap module creates an ad-hoc network with the following

SSID:	WiSnap-XX (where XX is the last two bytes of the devices MAC address)
Channel:	1
DHCP:	OFF
IP address:	169.254.1.1
Netmask:	255.255.0.0

With the ad-hoc jumper in place the above settings override the current saved configuration settings.

To enable ad-hoc mode from software:

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

```
set wlan join  4
set wlan ssid  my_ad-hoc_network
set wlan chan 1
```

Turn off DHCP and set the IP address and netmask so other devices know where to connect to the ad-hoc WiSnap. 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 begin statically in the same subnet as the ad-hoc device.

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

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

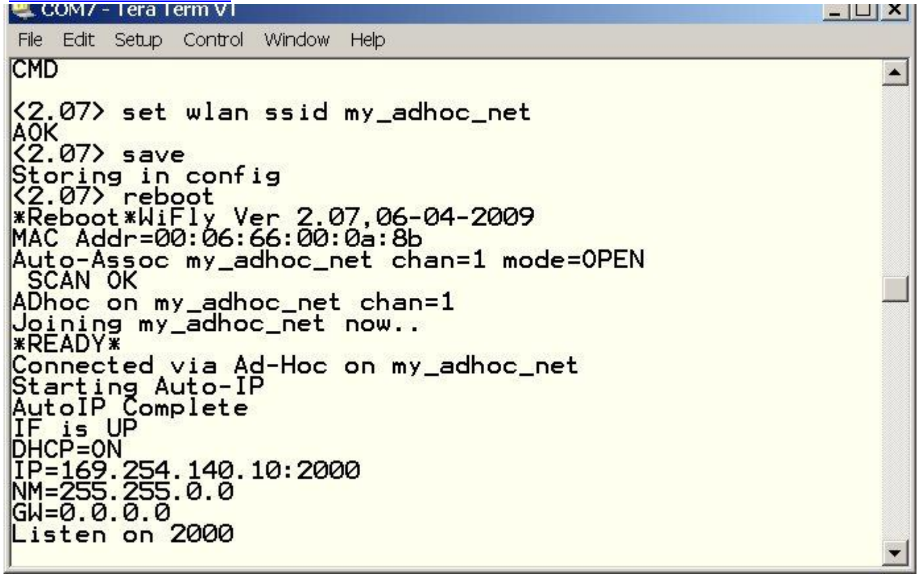
To associate with an ad-hoc network from another WiSnap device:

```
set wlan ssid  my_ad-hoc_network
reboot
```

Or alternatively, you can use the **join** command to associate with the ad-hoc network. Remember to disassociated using the **leave** command if you are previously associated to another network.

```
join my_ad-hoc_network
```

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



```
CMD

<2.07> set wlan ssid my_adhoc_net
AOK
<2.07> save
Storing in config
<2.07> reboot
*Reboot*WiFly Ver 2.07,06-04-2009
MAC Addr=00:06:66:00:0a:8b
Auto-Assoc my_adhoc_net chan=1 mode=OPEN
 SCAN OK
ADhoc on my_adhoc_net chan=1
Joining my_adhoc_net now..
*READY*
Connected via Ad-Hoc on my_adhoc_net
Starting Auto-IP
AutoIP Complete
IF is UP
DHCP=ON
IP=169.254.140.10:2000
NM=255.255.0.0
GW=0.0.0.0
Listen on 2000
```

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 ad-hoc network using the ping command.

```
ping 169.254.1.1 10
```

To use associate with the WiSnap ad-hoc 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 ad-hoc network.

Note: Once associated with the ad-hoc 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 ad-hoc network you can open a connection or telnet window as you would with an enterprise connection.
Note: The module does not support ad-hoc and enterprise network modes simultaneously.

15 Analog Sensor Capability

The WiSnap has 8 analog inputs that can be driven with 0 to 1.2VDC. Warning: over driving these inputs will cause permanent damage. The analog inputs are sampled and the digital value read by using the **show q <channel>** command.

The hardware specifications on the analog input is:
Input voltage range: 0 - 1.2V, 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 we 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, put exactly 200mV on analog input 4. then read channel 4 and compute the offset. Let us assume for example, if you read 210mv you would know that the offset is +10mv. When you read input 5 you would add 10mv to the result.

To read a sensor pin, send the following command:

show q <channel>

Channel is the analog input from 0 to 7. The value for the input channel you 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

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 option format <mask> command.

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>

Valid values are:

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 degrees 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.

16 Default Configuration Settings

AD-HOC PARAMETERS

Beacon	100 (milliseconds)
Probe	5 (retries before declaring ad-hoc 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 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

Join timer/WPA timer	1000
Replacement char	$ (0x24)
Device ID	WiSnap<DEVICEID>M1
Password	“” (blank-not used at this time)

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	WiSnap-XX (XX is the final two bytes of the devices MAC address)
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

16.1 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.

17 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	500
Ready	Load configuration and Initialize application	30
	Total time to READY	600
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-???

18 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 WiSnap should work with any standard Access Point. We have tested the WiSnap 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

- AD-HOC MODE (Apple iPhone, Microsoft Windows PC with XP, Vista , Ubuntu Linux)

19 Release Notes

19.1 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.**

19.2 Current Firmware features and fixes

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 WiSnap loses association the WiSnap will now closes the connection. The *CLOS* response will now appear when the connection is terminated by the WiSnap. 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 WiSnap declares "AP is lost" sets the interface to down state, and enters the association process. Previously the WiSnap would not detect that the AP association was lost until the AP became available again, or the WiSnap was power cycled or rebooted.

- **AD-HOC mode** The command **set ad-hoc probe x** is now used to set a threshold for the number of consecutive missed probe responses allowed before declaring "AD-HOC is Lost" and setting the network interface to be down. Default is 5 probes. A setting of **set ad-hoc probe 0** will disable this function. Some Ad-hoc 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 WiSnap 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 WiSnap 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), WiSnap 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 15

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 16.1

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 WiSnap 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 10

Known Issues

WiSnap Module has trouble associating with some 802.11.n access points. The module will crash and reboot repeatedly. We have seen this behaviour 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 WISNAP 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
- Ad-hoc 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- autosleep timer for UDP mode is not implemented.

Fixes since fimware version 2.05

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

SerialIO.com reserves the right to make corrections, modifications, and other changes to its products, documentation and services at any time. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

SerialIO.com assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using SerialIO.com components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

SerialIO.com products are not authorized for use in safety-critical applications (such as life support) where a failure of the SerialIO.com product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use.