

# The User Guide of ZD1211 USB Linux Driver

## Amendment History

Release Date	Release Version	Release Person	Release Description
2006.06.16	1.9	MingZhou Cai	Special mode for Embedded station chariot test
2006.05.25	1.8	MingZhou Cai	Some revision of incorrect information
2006.05.15	1.7	MingZhou Cai	Turbo Mode Support
2005.11.25	1.6	MingZhou Cai	Add Customize PID/VID Section
2005.09.15	1.5	MingZhou Cai	Add Dual band operation
2005.08.31	1.4	MingZhou Cai	Revise Continuous Tx Operation
2005.08.08	1.3	MingZhou Cai	Modification of compile process
2005.08.01	1.2	MingZhou Cai	Continuous Tx Operation
2005.06.15	1.1	MingZhou Cai	Add setRegion operation
2005.06.06	1.0	MingZhou Cai	This document is based on ZD1212 User Guide. Descriptions of unsupported operation is removed.



# 1. Introduction

Because more and more people install the Linux operating system in the desktop and notebook, we provide the Linux solution of our ZD1211 802.11a+b+g Wireless LAN Card. ZD1211 provides 802.11 a+b+g wireless solution for USB interface. In our ZD1211 solution, we can run in the Infrastructure (Managed), Ad-hoc or AP (Master) modes. One can easily change these modes. This document is intended to describe how to setup and how to use ZD1211 under the Linux operating system.

## 1.1 Requirements:

1. Kernel 2.4.20+ . The driver had successfully built in Redhat9, Fedora Core2,3 ,and Debian 3.1. You can easily build it without any problem
2. To build zd1211 you will need: Configured kernel source code for the kernel you are running. Ideally, Configured means that you have at least run 'make config', 'make menuconfig', or 'make xconfig'. If your platform is not SMP system, please don't config SMP supported, because when module loaded, this will make unresolved symbol.
3. Make sure your kernel usb 2.0 support is running
  - Use lsmod to check "ehci-hcd" module is loaded.
  - If host is not support usb 2.0, zd1211 will run under pure-b mode.

## 2. Building the device driver

In this section, we will describe how to build our ZD1211 Linux driver under the Linux operating system.

### 2.1 Package Extraction

```
tar zxvf ZD1211LnxDrv_XXXX.tar.gz
# where XXXX is the version number, such as 2_0_0_0
```

The first thing one should do is to uncompress this package by tar. After untaring this package, you can see the source files. One should change directory into this directory for proceeding the next step.

### 2.2 Build and install the driver

The package contains drivers for ZD1211 and ZD1211B. If you doesn't have specified request, both of them will be installed.

Under the extracted directory, there is a Makefile in it. Because our driver can support for kernel 2.4 and kernel 2.6, there are two sets of rule in the Makefile. One has to modify the Makefile according to the path of “**kernel source tree**” and the version of the kernel in your system. In the Makefile, you may see the following statements,

```
# if the kernel is 2.6.x, turn on this
#KERN_26=y
#KERNEL_SOURCE=/usr/src/linux-2.6.7
# if the kernel is 2.4.x, turn on this
KERN_24=y
KERNEL_SOURCE=/usr/src/linux-2.4.20-8
```

If you want to build the kernel under the kernel of 2.4.x, one has to set the variable KERN\_24=y and comment the KERN\_26=y like that as the example above and modify the variable KERNEL\_SOURCE to the path which you install the kernel source. After doing these things, one just need to type the “*make*”, and the driver module will be generated and installed.

### 2.3 Install individual driver

If you only need driver of ZD1211 or ZD1211B, you can issue :

```
make clean  
make ZD1211REV_B=0 (0 for ZD1211, 1 for ZD1211B)  
make ZD1211REV_B=0 install (0 for ZD1211, 1 for ZD1211B)
```

to install the driver.

## 2.4 Build the debugging tool

There are two debugging tools in this package, “*apdbg*” and “*menudbg*”. Run “*make debug*” to compile them both. If you don’t have the ncurses library, you may get some error messages while compiling *menudbg*. You can ignore it and get *apdbg* only.

## 3. Getting Start

### 3.1 Load the driver

Generally, the driver is automatically loaded when the zd1211 dongle inserts. If not, one can use the `modprobe -v zd1211(or zd1211b)` to load our driver. In order to check whether our driver is loaded successfully, one can use the “*lsmod*” for this check. If our driver is loaded successfully, the following messages should be seen

```
...  
zd1211 183576 0 (unused)
```

```
...
```

Please note that the 183576 may not be the same as that in your system.

### 3.2 Open the network interface

In our driver, we will stop all the commands until the network interface assigned to us is opened. You can open the network interface by the following command

```
]$ ifconfig ethX up  
or  
]$ ifconfig ethX <IP address>
```

### 3.3 Configure the Wireless settings

In our driver, we support the wireless extension commands to control our driver.

#### PARAMETERS:

##### *ssid :*

Set the ESSID (or Network Name - in some products it may also called Domain ID). The ESSID is used to identify cells which are part of the same virtual network.

Examples:

```
iwconfig ethX ssid <ESSID>
```

##### *mode:*

Set the operation mode of our device.

Examples:

```
iwconfig ethX <mode>
```

mode:

- Managed (Infrastructure Station mode)
- Ah-hoc (Ad hoc mode)
- Master (Access Point mode)

##### *channel:*

Set the channel in the device.

Example:

```
iwconfig ethX channel <channel>
```

The channel can vary from 1 to 14. One should note that, the set channel command will **NOT** work under the Managed (infrastructure mode). Because in the in this mode, the channel should change to channel of the AP we want to associate.

***rts[\_threshold]:***

Set the RTS Threshold.

Example:

```
iwconfig ethX rts 250
```

***frag[\_threshold]:***

Set the Fragmentation Threshold.

Example:

```
iwconfig ethX frag 512
```

***key/enc[ryption]:***

Used to manipulate encryption or scrambling keys and encryption mode. To set the current encryption key, just enter the key in hex digits as XXXX-XXXX-XXXX-XXXX or XXXXXXXX. To set a key other than the current key, append *[index]* to the key itself. You can also enter the key as an ASCII string by using the *s:* prefix. To change which key is the current active key, just enter *[index]* (without entering any key value). *off* and *on* disable and reenale encryption, *open* set the system in open mode (accept non-encrypted packets) and *restricted* discard non-encrypted packets.

Examples :

```
iwconfig ethX key 0123-4567-89 [1]
```

```
iwconfig ethX key [1] open
```

```
iwconfig wlan0 key off
```

***power:***

Used to manipulate the power management scheme mode.

Examples:

```
iwconfig ethX power on # Turn on power saving mode
```

```
iwconfig ethX power off #Turn off power saving mode
```

## 3.4 Private commands

Except for commands support for wireless extension, we also define some commands for us to set parameters to our driver. One can use the “*iwpriv*” for this purpose.

### 3.4.1 Set authentication type

One can set the authentication to our driver by the following command:

```
iwpriv ethX set auth <Auth Type>
```

0: Open System

1: Shared Key

**Be aware that shared key authentication requires a WEP key.**

### 3.4.2 Set preamble type

One can set the preamble type to our driver by the following command:

```
iwpriv ethX set preamble <Preamble Type>
```

0: Long preamble

1: Short preamble

### 3.4.3 Get preamble type

One can get the preamble type of our driver by the following command:

```
iwpriv ethX get_preamble
```

### 3.4.4 Set MAC mode

Because the ZD1211 is a+b+g solution, we support the PURE\_A, PURE B, PURE G and Mixed mode in our driver. One can use the following command to change the MAC mode in our driver.

```
iwpriv ethX set_mac_mode <MAC mode>
```

1: Mixed Mode

2: Pure G Mode

3: Pure B Mode

4: Pure A Mode

### 3.4.5 Get MAC mode

One can get the MAC mode of our driver by the following command

```
iwpriv ethX get_mac_mode
```

### 3.4.6 Connect to the given Access Point

One can associate with the given Access Point with a given Cell Number by the following command.

```
iwpriv ethX connect <Cell Number>
```

The Cell Number is got from the site survey operation by the doing “iwlist” command. We recommend that user uses the following scenario under the Managed (Infrastructure) or Adhoc mode. One can first do the site survey command by the following command:

```
iwlist ethX scanning
```

Then, associate with the AP with the Cell number got from the iwlist command.

```
iwpriv ethX connect <Cell Number>
```

### 3.4.7 Dynamical Region Setting(For 802.11b/g only)

From ver 1.5, you can dynamically change the region settings. With different regions, the allowed channels are different. The private commands to get/set region information includes :

1. **get\_Region** : To get the current region setting

```
iwpriv ethX connect <Cell Number>
```

2. **set\_Region** : To set the region

```
iwpriv ethX connect <Cell Number>
```

RegionID	Region String	Channel	Countries
1	USA	1-11	USA,Canada,Argentina,Brazil,Ukraine,China,HongKong ,Korea,NewZealand
2	Taiwan/Europe	1-13	Taiwan, Europe, Spain, Austria, Belgium, Switzerland, Australia
3	France	10-13	France, Singapore
4	Japan	1-14	Japan
5	Israel	3-9	Israel
6	Mexico	10-11	Mexico

The table of region id and region string

### 3.4.8 Activate MIC Fail Counter Measure Feature

The feature will disable the network connection for one minutes when you get two MIC fail frame in one minute. The feature is initially disabled. You can activate the feature by using the *apdbg* tool.

```
apdbg ethX s_mic_cnt 0 1
```

Or inactivate it.



```
apdbg ethX s_mic_cnt 0 0
```

Or get its current state

```
apdbg ethX g_mic_cnt 0 0
```

## 3.5 Set up IP address

If you use the RedHat distribution Linux, you can edit the `/etc/sysconfig/network-scripts/ifcfg-ethX` or edit the `/etc/network/interfaces` under the Debian to set up the IP address on booting process. Or one can use the `netconfig` command for ip address setting.

We provide two types setting in the following examples. One is to assign a fix IP address, netmask, and default gateway. Another is to get IP configuration from a DHCP server.

### 3.5.1 Fixed Setting

# This is an example of fixed IP setting

```
DEVICE='eth0'  
IPADDR='192.168.2.98'  
NETMASK='255.255.255.0'  
NETWORK='192.168.2.0'  
BROADCAST='192.168.2.255'  
ONBOOT='yes'  
GATEWAY='192.168.2.254'
```

### 3.5.2 Get IP setting from DHCP

# This is an example of getting ip from DHCP server.

```
DEVICE='eth0'  
BOOTPROTO='dhcp'  
ONBOOT='yes'
```

### 3.5.3 Setting Access Point

#### 3.5.3.1 The typical setting procedure:

```
iwconfig ethx mode master           // Set to AP mode  
iwconfig ethx essid ssid           // Set ssid  
iwpriv ethx set_mac_mode mac_mode //Ref section 3.4.4 Set MAC mode  
iwconfig ethx channel channel#     // Available channel # is 1,2,3,4..etc
```

## 3.6 Working with Linux WPA supplicant

**Note:** I do the following procedure in Fedora Core2, for other distribution package, you may need install additional libraries(ex. openssl) required to build the wpa supplicant..

### 3.6.1 Setup the Linux wpa supplicant

- Copy wpa\_supplicant.tar.gz file into a subdirectory on Linux system(e.g:/root).
- Unzip it by using command:

```
tar zxvf wpa_supplicant.tar.gz
```

Then, a subdirectory of wpa\_supplicant/ will be created under the current directory.

- Enter subdirectory wpa\_supplicant/
- Delete the original .config file by:

```
rm -f .config
```

- Edit Makefile, make sure the following statements in mkconfig: section:

```
echo CONFIG_IEEE8021X_EAPOL=y >> .config
echo CONFIG_EAP_MD5=y >> .config
echo CONFIG_MSCHAPV2=y >> .config
echo CONFIG_EAP_PEAP=y >> .config
echo CONFIG_EAP_TLS=y >> .config
echo CONFIG_DRIVER_WEXT=y >> .config
echo CONFIG_WIRELESS_EXTENSION=y >> .config
echo CONFIG_DRIVER_ZYDAS=y >> .config
```

- Create the new .config file by:

```
make mkconfig
```

- Now, we can build the Linux wpa supplicant by entering following command:

```
make
```

After make process completed, A executable file wpa\_supplicant created.

- To create a WPA PSK connection, please modify the configuration file (For detailed description , you can refer to the original sample configuration file: wpa\_supplicant.conf) wpa\_supplicant\_psk.conf to meet wpa-psk test condition.

Sample settings for wpa-psk:

```
network={
    ssid="wrt55ag"
    proto=WPA
    key_mgmt=WPA-PSK
    pairwise=CCMP TKIP
    group=CCMP TKIP WEP104 WEP40
    psk="12345678"
    priority=2
}
```

Similarly, for wpa eap-tls and wpa peap, its sample setting block:  
For WPA EAP-TLS

```
network={
    ssid="wrt55ag"
    proto=WPA
    key_mgmt=WPA-EAP
    pairwise=CCMP TKIP
    group=CCMP TKIP WEP104 WEP40
    eap=TLS
    identity="Administrator@zydas.com.tw"
    ca_cert="/etc/cert/fluffy.pem"
    client_cert="/etc/cert/id.pem"
    private_key="/etc/cert/id_key.pem"
    private_key_passwd="password"
    priority=2
}
```

For NonWPA 802.1X network, just change key\_mgmt field from WPA-EAP to IEEE8021X.

Note1:

The fluffy.pem is created by:

- ~~openssl pkcs7 -in fluffy.p7c -inform DER -print\_certs -outform PEM -out fluffy.pem~~
- openssl pkcs12 -in fluffy.pfx -passin pass:password -out fluffy.pem -cacerts -nokeys

The id\_key.pem is created by

- > openssl pkcs12 -in fluffy.pfx -passin pass:password -passout pass:password -out id\_key.pem -nocerts

The id.pem is created by

- > openssl pkcs12 -in fluffy.pfx -passin pass:password -out id.pem -nokeys

Note2:

You can run openssl utility (Included in openssl.zip) in Microsoft Windows OS.

Note3:

The detailed description, please refer to CertConvReadme.txt. (Located in ln\_x\_wpa\_suppllicant.tar.gz)

For WPA PEAP

```
network={
  ssid="example"
  key_mgmt=WPA-EAP
  eap=PEAP
  identity="jhsieh"
  password="jhsieh"
  ca_cert="/etc/cert/fluffy.pem"
  phase1="peaplabel=0"
  phase2="auth=MSCHAPV2"
  priority=10
}
```

- After modifying, use the following command to setup WPA connection.

If the zd1211 is not open yet, please open it firstly by command:

```
ifconfig eth1 up <IP address of the network interface>
```

After network interface is opened, enter the command to build wpa psk connection:

```
./wpa_suppllicant -ieth1 -c wpa_suppllicant_psk.conf -d -D zydas
```

To build wpa eap-tls

```
./wpa_suppllicant -ieth1 -c wpa_suppllicant_tls.conf -d -D zydas
```

To build wpa peap:

```
./wpa_suppllicant -ieth1 -c wpa_suppllicant_peap.conf -d -D zydas
```

```

note:
@-i: interface name: eth1
@-c: Configuration file: wpa_supplicant_psk.conf
@-D: The name of network interface.
You will see the following message if wpa-psk connection is built successfully.
...
WPA: Sending EAPOL-Key 2/2    ---> The Group handshake is about to finish.
...
EAPOL: SUPP_PAE entering state SUCCESS
EAP: EAP entering state SUCCESS
EAPOL: SUPP_PAE entering state AUTHENTICATED
EAPOL: SUPP_BE entering state IDLE

```

Note of wpa supplicant operation issue:

The WPA supplicant should be keep running during connection. If you press Ctrl-C to stop the wpa supplicant, it will also close the network interface card by "zd1205\_close" call back routine. So you have to issue ifconfig ethx up again before using the network interface card.

### 3.7 Customization of VID/PID Table

Someone may want to drive any NIC with ZD1211 chipset. But, if its VID/PID isn't listed in the driver, the linux kernel won't load the zd1211 driver to serve the device. You can add the device's VID/PID to the driver source. Then, the kernel will load driver next time you plug the USB dongle.

In the source file, src/zdusb.c, A static structure, zd1211\_ids, stores the VID/PID list of supported products. You can add your dongle's VID/PID here. Then, remake, install the driver and plug the dongle. You will see it works now.

### 3.8 Special Mode for Embedded Station Chariot Test

In an embedded system, mostly, there is no NetIQ chariot endpoint, for WiFi testing, to test performance. With this mode, we can bridge the Ethernet and WLAN. The PC behind the Ethernet runs endpoint and testing data go through the station's wireless interface to measure the performance.

The mechanism is to specify the MAC address for association with AP. Generally, we associate with AP using the station's MAC address. If we bridge the wired and wireless interfaces, data from wired interface will be dropped by AP. Because PC's MAC address is filled in source address of data frames. AP drops the frames because the source address is not in associated stations list. If we can use AP's address for association, we can run chariot on PC to test station wireless interface performance. This is just like what we do in AP mode.

Someone may say that "Can't I just set station's MAC as PC's via ifconfig ethX hw ether". Yes, perhaps, you can. But most bridge will drop the data when the source address is identical to its interfaces'. Therefore, we must let the address for association and

address of Station different.

The command to specify association address is very simple :

```
ifconfig ethX down  
ifconfig ethX hw ether 01XXXXXXXXXX ;  
ifconfig ethX up
```

You must set the first byte = 01 to indicate this is for association address assignment only. The remainder XXXXXXXXXXXX is byte1-byte5 of PC's MAC address. Now, you can set up a bridge and add both wired and wireless interface. Then, you can try to ping to test the connectivity.

## 4. ZyDAS Turbo Mode

This chapter describes how to enhance the performance by enabling the ZyDAS turbo mode. Turbo mode is a composition of **Burst Mode** and **Large Packet Mode**.

You may know that the so-call “Turbo Mode” always has compatibility problems. In order to achieve the highest performance, you need to use ZyDAS’s product in both AP and Station. The Turbo mode is available on ZyDAS ZD1211B/ZD1212B with Linux driver only in this moment. Currently, ZD1211 supports it in **Infrastructure Mode** only.

You could enable the both to achieve maximal performance.

### 4.1 Enable Burst Mode

To enable burst mode, you need to issue the following commands.

```
iwpriv ethX lp_mode 4
```

To disable burst mode :

```
iwpriv ethX lp_mode 5
```

**In Infrastructure mode, You need to re-associate with the AP to update your Turbo Mode status .**

### 4.2 Large Packet mode

In Large Packet mode, small frames are merged into a large one. It can save inter-frame spacing time to raise the performance. The command to activate large packet mode is :

To Enable:

```
iwconfig ethX rts off  
iwpriv ethX lp_mode 1
```

To Disable:

```
iwpriv ethX lp_mode 0
```

The “enable” means to send with large packet enabling. The receiving capability is always on. For AP mode with `lp_mode = 1`, it sends large packet to a station only when the last frame from that station is a large packet. Therefore, AP can accept station capable

of large packet or not. **In Station mode, You need to re-associate with the AP to update your Turbo Mode status.**

You need to remind something in mind. Large packet needs to merge two frames into one. In other words, when there is only one frame in queue, it could take some time to wait another one. If the next one comes too late or not come, it will be sent by a expire checking timer. It could result in lower performance than normal mode. Our experience shows two traffic flows achieve best performance. Fragmentation isn't allowed when Large Packet is on.



## **5. Conclusion**

This document doesn't explain how to setup the wireless LAN environment in detail. One may get some problems when setting up the wireless LAN environment. If you have any question about how to set up the environment, you can send an e-mail to us or find the solution on the network.