

---

**How To Work WiFi (WILC1000)**

---

**Atmel SmartConnect**

---

**Introduction**

---

This document describes how to work WiFi on Linux console.

## Prerequisites

---

## Table of Contents

---

<b>Introduction</b>	<b>1</b>
<b>Prerequisites</b>	<b>2</b>
<b>Table of Contents</b>	<b>3</b>
<b>Icon Key Identifiers</b>	<b>5</b>
<b>1 Overview Stack</b>	<b>6</b>
<b>2 Load/Unload Kernel Driver module</b>	<b>7</b>
2.1 Detecting ATWILC1000 SDIO Card Module	7
2.2 Loading ATWILC1000 Driver Module	7
2.3 Unloading ATWILC1000 SDIO Card Module	7
<b>3 How to Work Station Mode</b>	<b>8</b>
3.1 Write wpa_supplicant.conf	8
3.2 Start WPA Supplicant Service	8
3.3 Connect with AP	9
3.4 Obtain IP	11
3.5 Check the connected status	11
3.6 Disconnect with AP and Release	12
3.7 Example Scenario	12
3.8 Example Command with iw utility	13
<b>4 How to Work Access Point Mode</b>	<b>14</b>
4.1 Write hostapd.conf	14
4.2 Start Hostapd service	15
4.3 Allocate IP	16
4.4 Stop Soft AP mode and release	17
4.5 Example Scenario like internet sharing with wire LAN	17
4.6 Example Scenario with EAP server	18
<b>5 How to Work Peer to Peer(P2P)</b>	<b>20</b>
5.1 Write p2p_supplicant.conf	20
5.2 Running as P2P mode	20
5.3 Allocate IP	20
5.4 Connect with Peer	21
5.5 Disconnect with Peer and Release	22
<b>6 How to Work Concurrent Mode (Both Station and AP)</b>	<b>23</b>
6.1 Start Station mode in Concurrency	23
6.2 Start Access Point mode in Concurrency	23
6.3 Stop Concurrent mode	25
6.4 Internet Sharing Example on Concurrency	25
<b>7 Benchmark using iPerf</b>	<b>28</b>
7.1 Testing downlink	28
7.2 Testing uplink	29
<b>8 How to use Dynamic Log Message</b>	<b>30</b>
8.1 Debugfs Mount	30

8.2	Change Debug Level.....	30
<b>9</b>	<b>Revision History .....</b>	<b>31</b>

## Icon Key Identifiers

---



### INFO

Delivers contextual information about a specific topic.



### TIPS

Highlights useful tips and techniques.



### TO DO

Highlights objectives to be completed.



### RESULT

Highlights the expected result of an assignment step.



### WARNING

Indicates important information.



### EXECUTE

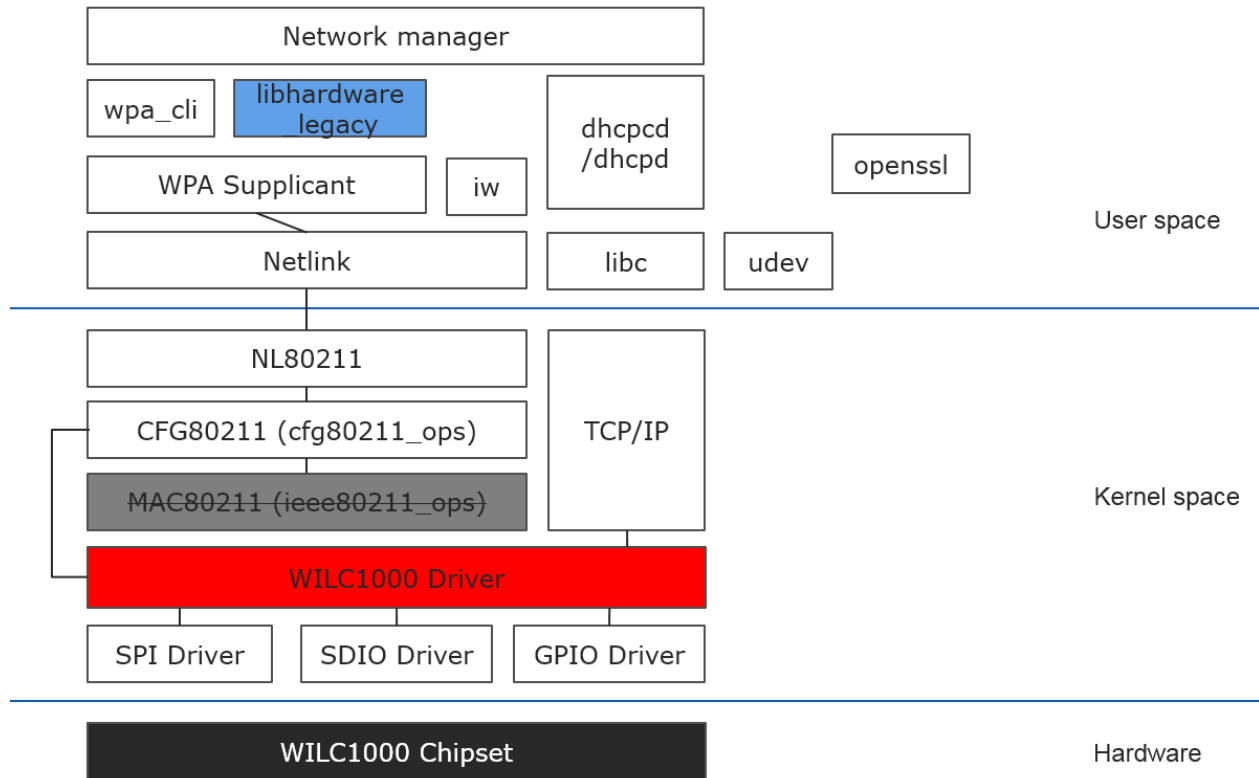
Highlights actions to be executed out of the target when necessary.

# 1 Overview Stack

Linux environment requires some mandatory package each space basically to run WILC1000. User should include the stacks in built binary as well as WILC1000 driver.

Below picture will help you for understanding.

## Dfd



Because WILC1000 chipset is the full MAC device, software stack of MAC80211 isn't required.

**wpa\_supplicant** is a free software implementation of an IEEE 802.11i supplicant for Linux, FreeBSD, NetBSD, QNX, AROS, Microsoft Windows, Solaris, OS/2 (including eComStation) and Haiku. In addition to being a fully featured WPA2 supplicant, it also implements WPA and older wireless LAN security protocols. Features include:

- WPA and full IEEE 802.11i/RSN/WPA2
- WPA-PSK and WPA2-PSK ("WPA-Personal", pre-shared key)
- WPA with EAP ("WPA-Enterprise", for example with RADIUS authentication server)
- key management for CCMP, TKIP, WEP (both 104/128- and 40/64-bit)
- RSN: PMKSA caching, pre-authentication

Ref : <https://w1.fi/>

## 2 Load/Unload Kernel Driver module

This chapter explains how to load and unload WILC1000 Driver on kernel memory to run WiFi.

### 2.1 Detecting ATWILC1000 SDIO Card Module

If user insert the ATWILC1000 card into SD Card slot, host platform recognize the card with the following line or equivalent.

```
mmc0: new high speed SDIO card at address 0001
```

### 2.2 Loading ATWILC1000 Driver Module

Kernel drivers normally are located as kernel version name in /lib/modules tree of filesystem.

Input the following command with proper version name on the terminal to load driver module.

```
$ insmod /lib/module/$(uname -r)/kernel/driver/net/wireless/atmel/wilc1000/wilc1000.ko
```

```
IN INIT_FUNCTION
*** WILC1000 driver VERSION=[XX.X.X] ***
Driver Initializing success
```

User can check if the module rightly is loaded using 'lsmod' command.

```
$ lsmod
```

Module	Size	Used by	Not tainted
wilc1000	224005	0	

### 2.3 Unloading ATWILC1000 SDIO Card Module

```
$ rmmod wilc1000
```

```
Module_exit Done.
```

## 3 How to Work Station Mode

Station mode is normal Wi-Fi operation that is connecting to a given AP. This chapter explains how to connect with the given AP.

### 3.1 Write wpa\_supplicant.conf

A configuration file should be prepared for wpa\_supplicant service. The file name can be rename by user. But the contents have the basic rules of grammar in the file and if there isn't a specific reason, place the file in /etc folder.

Visit [https://w1.fi/cgiit/hostap/plain/wpa\\_supplicant/wpa\\_supplicant.conf](https://w1.fi/cgiit/hostap/plain/wpa_supplicant/wpa_supplicant.conf) page to get the detail options.

```
$ vi /etc/wilc_wpa_supplicant.conf
```

Ex)

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
```



#### TIPS

If user already know the information of the given AP and filled the information in conf file, when wpa\_supplicant service start, WILC1000 will try to connect with the AP automatically.

Ex) Connecting with wpa/wpa2 AP

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1

network={
    ssid="User_AP"
    psk="password"
}
```

Ref : [http://linux.die.net/man/5/wpa\\_supplicant.conf](http://linux.die.net/man/5/wpa_supplicant.conf)

### 3.2 Start WPA Supplicant Service

Execute wpa\_supplicant with the following command on the terminal. The wpa\_supplicant will automatically perform the background scan with "ifconfig wlan0 up" command.



```
$ wpa_supplicant -B -Dnl80211 -iwlan0 -c/etc/wilc_wpa_supplicant.conf
```

```
DBG [wlan_init_locks: 1810]Initializing Locks ...
DBG [linux_to_wlan: 1881]Linux to Wlan services ...
DBG [wilc_wlan_init: 2938]Initializing WILC_Wlan ...
[wilc sdio]: chipid (001002b0)
DBG [wilc_wlan_firmware_download: 2282]Downloading firmware size = 142676 ...
DBG [linux_wlan_firmware_download: 1403]Freeing FW buffer ...
DBG [linux_wlan_firmware_download: 1404]Releasing firmware
DBG [linux_wlan_firmware_download: 1408]Download Succeeded
DBG [linux_wlan_start_firmware: 1342]Starting Firmware ...
DBG [linux_wlan_start_firmware: 1353]Waiting for Firmware to get ready ...
DBG [linux_wlan_start_firmware: 1379]Firmware successfully started
...
```



**WARNING** Because NL80211 may not work as the status of Kernel version or WPA supplicant, 'nl80211' option can be replaced to 'wext' option.

```
# wpa_supplicant -B -Dwext -iwlan0 -c/etc/wilc_wpa_supplicant.conf
```

## 3.3 Connect with AP

### 3.3.1 To an unencrypted AP

The following commands demonstrate how to scan and connect to the AP.

```
$ wpa_cli -p/var/run/wpa_supplicant ap_scan 1
$ wpa_cli -p/var/run/wpa_supplicant add_network
$ wpa_cli -p/var/run/wpa_supplicant set_network 0 ssid '"User_AP"'
$ wpa_cli -p/var/run/wpa_supplicant set_network 0 key_mgmt NONE
$ wpa_cli -p/var/run/wpa_supplicant select_network 0
```

Log



**TIPS** Change "User\_AP" with the SSID of the desired AP.

### 3.3.2 To WPA protected AP

To connect to AP that uses WPA or WPA2 and TKIP or AES, the following commands demonstrate how to scan and connect to the protected AP.

```
$ wpa_cli -p/var/run/wpa_supplicant ap_scan 1
$ wpa_cli -p/var/run/wpa_supplicant add_network
$ wpa_cli -p/var/run/wpa_supplicant set_network 0 ssid '"User_AP"'
$ wpa_cli -p/var/run/wpa_supplicant set_network 0 psk '"password"'
$ wpa_cli -p/var/run/wpa_supplicant select_network 0
```



#### TIPS

Change “User\_AP” and “12345678” with SSID and password of desired AP.

### 3.3.3 To WEP protected AP

To connect to AP that uses WEP40 or WP104, the following commands demonstrate how to scan and connect to the protected AP.

```
$ wpa_cli -p/var/run/wpa_supplicant ap_scan 1
$ wpa_cli -p/var/run/wpa_supplicant add_network
$ wpa_cli -p/var/run/wpa_supplicant set_network 0 ssid '"User_AP"'
$ wpa_cli -p/var/run/wpa_supplicant set_network 0 key_mgmt NONE
$ wpa_cli -p/var/run/wpa_supplicant set_network 0 wep_key0 1234567890
$ wpa_cli -p/var/run/wpa_supplicant set_network 0 wep_tx_keyidx 0
$ wpa_cli -p/var/run/wpa_supplicant set_network 0 auth_alg SHARED
$ wpa_cli -p/var/run/wpa_supplicant select_network 0
```



#### TIPS

Change “User\_AP” and “1234567890” with SSID and ASCII (or Hex) of desired AP.

### 3.3.4 With WPS[WiFi Protected Setup] function

Wi-Fi Protected Setup™ is an optional certification program based on technology designed to ease the setup of security-enabled Wi-Fi® networks in home and small office environments. Wi-Fi Protected Setup supports methods (pushing a button, entering a PIN, or using NFC) that are familiar to most consumers to configure a network and enable security. WILC1000 supports pushing a button and entering a PIN.

Ref : <https://www.youtube.com/watch?v=74QBV9BBXVc&feature=youtu.be>

- Pushing a button

```
$ wpa_cli -p/var/run/wpa_supplicant wps_pbc any
```

After the connection is completed successfully with the given AP, “wlc\_wpa\_supplicant.conf” file is updated like below. So, when user restart wpa\_supplicant service next time, wlc1000 will connect to the saved AP automatically.

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1

network={
```

```
ssid="Given_AP"  
key_mgmt=NONE  
auth_alg=OPEN  
}
```

- Entering a Pin

```
$ wpa_cli -p/var/run/wpa_supplicant wps_pin any 12345670
```

### 3.4 Obtain IP

Network device(wlan0) should obtain IP for packet communication on the connected networks. User chooses one of the two for IP allocation.

#### 3.4.1 If clients want to configure manually IP

User know the network information in advance and should set the value.

```
$ ifconfig wlan0 xxx.xxx.xxx.xxx netmask xxx.xxx.xxx.xxx  
$ route add default gw xxx.xxx.xxx.xxx wlan0  
$ echo "nameserver xxx.xxx.xxx.xxx" > /etc/resolv.conf
```

#### 3.4.2 If IP of clients want to be allocated by a given AP automatically

If the given AP supports DHCP server, network device automatically obtains IP.

```
$ dhcpcd wlan0 &
```

or

```
$ udhcpc -i wlan0
```

### 3.5 Check the connected status.

Check the following command to see that the connection is established.

```
$ wpa_cli status
```

```
bssid=88:9b:39:f3:d0:4d
ssid=User_AP
id=0
mode=station
pairwise_cipher=NONE
group_cipher=NONE
key_mgmt=NONE
wpa_state=COMPLETED
ip_address=XXX.XXX.XXX.XXX
address=00:80:c2:b3:d7:4d
```

### 3.6 Disconnect with AP and Release

```
$ wpa_cli -p/var/run/wpa_supplicant disconnect
$ wpa_cli -p/var/run/wpa_supplicant remove_network 0
$ start-stop-daemon -K -x /usr/sbin/dhccpd
$ killall wpa_supplicant
```

### 3.7 Example Scenario

Below example is how to change to “User\_AP2” after connection with “User\_AP1”.

Load driver	<code>insmod wilc1000.ko</code>
Execute wpa_supplicant	<code>wpa_supplicant -B -Dnl80211 -iwlan0 -c/etc/wilc_wpa_supplicant.conf</code>
Register User_AP1 information	<code>wpa_cli -p/var/run/wpa_supplicant ap_scan 1</code> <code>wpa_cli -p/var/run/wpa_supplicant add_network</code> <code>wpa_cli -p/var/run/wpa_supplicant set_network 0 ssid '"User_AP1"'</code> <code>wpa_cli -p/var/run/wpa_supplicant set_network 0 key_mgmt NONE</code>
Connect with User_AP1	<code>wpa_cli -p/var/run/wpa_supplicant select_network 0</code>
Register User_AP2 information	<code>wpa_cli -p/var/run/wpa_supplicant ap_scan 1</code> <code>wpa_cli -p/var/run/wpa_supplicant add_network</code> <code>wpa_cli -p/var/run/wpa_supplicant set_network 1 ssid '"User_AP2"'</code> <code>wpa_cli -p/var/run/wpa_supplicant set_network 1 psk '"12345678"'</code>
Disconnect with User_AP1	<code>wpa_cli -p/var/run/wpa_supplicant disconnect</code>
Connect with User_AP2	<code>wpa_cli -p/var/run/wpa_supplicant select_network 1</code>
Unregister User_AP1 Information	<code>wpa_cli -p/var/run/wpa_supplicant remove_network 0</code>
Disconnect with User_AP2	<code>wpa_cli -p/var/run/wpa_supplicant disconnect</code>
Unregister User_AP2 Information	<code>wpa_cli -p/var/run/wpa_supplicant remove_network 1</code>
Terminate wpa_supplicant	<code>killall wpa_supplicant</code>
Unload driver	<code>rmmmod wilc1000</code>

### 3.8 Example Command with iw utility

Visit <https://wireless.wiki.kernel.org/en/users/Documentation/iw> page to get the detail options of iw.

- Scanning

```
$ iw dev wlan0 scan
```

- To connect to an AP that has encryption disabled, where its SSID is foo:

```
$ iw wlan0 connect foo
```

- To connect to an AP that uses WEP, you can use:

```
$ iw wlan0 connect foo keys 0:abcde d:1:0011223344
```

key must be [d:]index:data where  
'd:' means default (transmit) key  
'index:' is a single digit (0-3)  
'data' must be 5 or 13 ascii chars  
or 10 or 26 hex digits

- To determine if you are connected to an AP or not, you can use:

```
$ iw dev wlan0 link
Connected to XX:XX:XX:XX:XX:XX (on wlan0)
    SSID: foo
    Freq: 2437
    ...
```

- Example output when not connected to an AP:

```
$ iw dev wlan0 link
Not connected.
```

## 4 How to Work Access Point Mode

### 4.1 Write hostapd.conf

A configuration file should be prepared for hostapd daemon. The file name can be rename by user. But the text of the file have the basic rules of grammar as AP's security mode. If there isn't a specific reason, place the file in "/etc" folder.

Visit <http://w1.fi/cgiit/hostap/plain/hostapd/hostapd.conf> page to get the detail options.

- As an unencrypted AP

```
$ vi /etc/wilc_hostapd_open.conf
```

```
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ssid=wilc1000_SoftAP
dtim_period=2
beacon_int=100
channel=1
hw_mode=g
max_num_sta=8
ap_max_inactivity=300
```

- As WPA protected AP

```
$ vi /etc/wilc_hostapd_wpa.conf
```

```
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ssid=wilc1000_SoftAP
dtim_period=2
beacon_int=100
channel=6
hw_mode=g
max_num_sta=8
ap_max_inactivity=300
ieee80211n=1
auth_algs=1

##### WPA/WPA2 #####
wpa=3
wpa_passphrase=12345678
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP CCMP
rsn_pairwise=CCMP
```

- As WEP protected AP

```
$ vi /etc/wilc_hostapd_wep.conf
```

```
interface=wlan0
driver=n180211
ctrl_interface=/var/run/hostapd
ssid=wilc1000_SoftAP
dtim_period=2
beacon_int=100
channel=6
hw_mode=g
max_num_sta=8
ap_max_inactivity=300
ieee80211n=0
auth_algs=1

##### WEP #####
wep_default_key=0
wep_key0=1234567890
wep_key1="vwxyz"
wep_key2=0102030405060708090a0b0c0d
wep_key3=".2.4.6.8.0.23"
wep_key_len_unicast=5
wep_rekey_period=300
```

## 4.2 Start Hostapd service

**hostapd** is a user space daemon for wireless access point and authentication servers. hostapd will execute soft AP as user's configuration.

The following command demonstrates how to execute the hostapd.

```
$ hostapd -B /etc/wilc_hostapd_xxx.conf
```

```
Configuration file: /etc/wilc_hostapd_XXX.conf
rfkill: Cannot open RFKILL control device
loading firmware atmel/wilc1003_firmware.bin
set drv handle = ddd06000 , 0
Using interface wlan0 with hwaddr f8:f0:05:f1:33:3e and ssid "wilc1000_D4"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
...
```



### TIPS

Refer to 'wilc\_hostapd\_open.conf' for the details of unencrypted AP settings. Refer and use 'wilc\_hostapd\_wpa.conf' for WPA/WPA2 AP setting.

## 4.3 Allocate IP

Access Point should have IP to communicate with clients. Configure own IP as gateway with following command.

```
$ ifconfig wlan0 192.168.xxx.xxx
```

Clients may be the IP communication after users directly input IP as the way of 3.4.1 chapter.

### 4.3.1 If IP of clients want to be automatically allocated by AP

If clients want to be allocated from Soft AP or others, DHCP server should be executed in the background

- Set IP as gateway.

```
$ ifconfig wlan0 192.168.123.1
```



#### TIPS

Gateway IP is defined in the dhcpd.conf.

- Write the dhcpd.conf like the following.
  - /etc/dhcpd.conf

```
ddns-update-style none;
default-lease-time 600;
max-lease-time 7200;

option subnet-mask 255.255.255.0;
option domain-name-servers 168.126.63.1, 164.124.101.2; # DNS Server IP
option domain-name "sample.example"; # domain name

subnet 192.168.123.0 netmask 255.255.255.0 {
    range 192.168.123.100 192.168.123.110; # range ip
    option broadcast-address 192.168.123.255;
    option routers 192.168.123.1; # gateway ip
}
log-facility local7;
```



#### TIPS

Each value must be modified to fit the test environment.

- Start the DHCP Server

```
$ mkdir -p /var/lib/dhcp/
$ touch /var/lib/dhcp/dhcpd.leases
$ dhcpd wlan0
```

Now the clients can be connected to the ATWILC1000 device, which is running in AP mode.





## TIPS

Otherly, user can execute DHCP server after modifying “/etc/init.d/S80dhcp-server” script.

## 4.4 Stop Soft AP mode and release

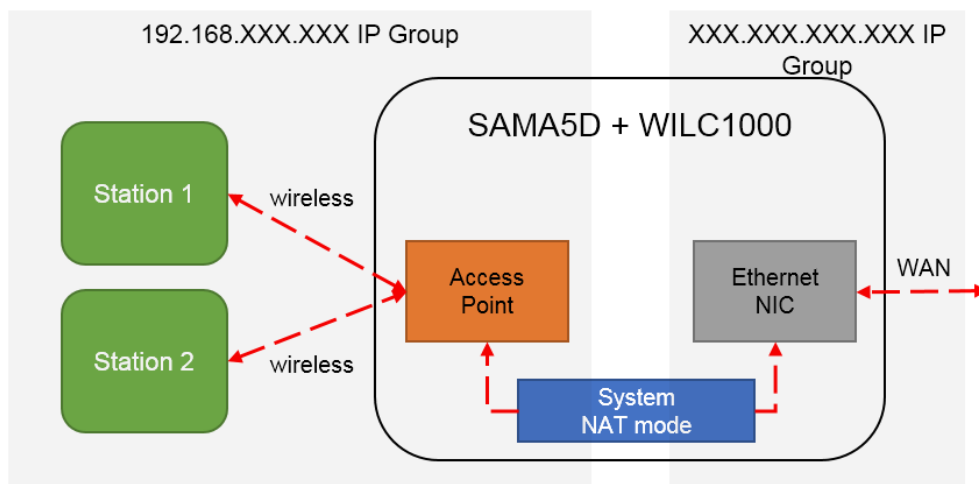
```

$ start-stop-daemon -K -x /usr/sbin/dhcpd
$ killall hostapd
$ rmmod wilc1000

```

## 4.5 Example Scenario like internet sharing with wire LAN

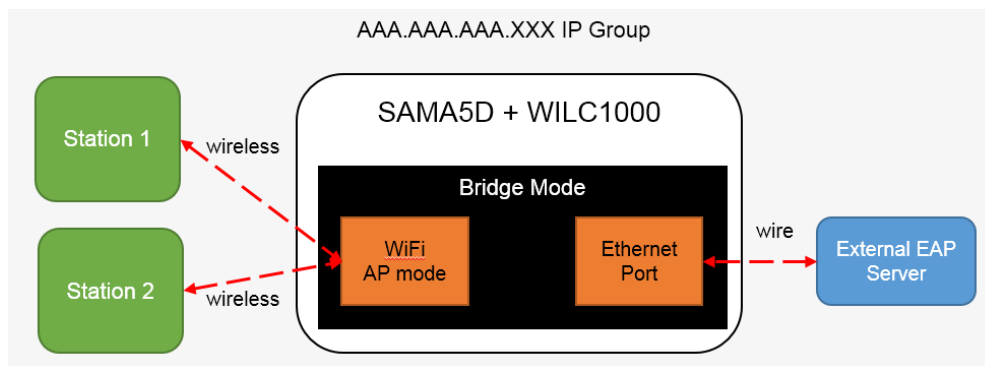
- Demanded package.
    - Kernel driver : Netfilter
    - User space application : iptables
- Ref Link : <https://github.com/linux4sc/wireless-driver/wiki/NAT#2-kernel-configurations>



Load driver	<code>insmod wilc1000.ko</code>
Execute hostapd	<code>hostapd -B /etc/wilc_hostapd_open.conf</code>
Set AP's IP	<code>ifconfig wlan0 192.168.123.1</code>
Execute DHCP server (optional)	<code>mkdir -p /var/lib/dhcp/</code> <code>touch /var/lib/dhcp/dhcpd.leases</code> <code>dhcpd wlan0</code>
Configure wire Ethernet card	<code>ifconfig eth0 up</code> <code>Allocate IP for eth0</code>
Execute NAT(iptables)	<code>echo 1 &gt; /proc/sys/net/ipv4/ip_forward</code>

	<pre>iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT iptables -A FORWARD -o wlan0 -i eth0 -j ACCEPT iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE</pre>
Clients can be connected with wilc1000 AP	Is doing
Terminate NAT (iptables)	<pre>iptables -F or iptables -D FORWARD 1 &amp;&amp; iptables -D FORWARD 2 echo 0 &gt; /proc/sys/net/ipv4/ip_forward</pre>
Unload Ethernet card	<code>ifconfig eth0 down</code>
Terminate DHCP server	<code>start-stop-daemon -K -x /usr/sbin/dhcpd</code>
Terminate hostapd	<code>killall hostapd</code>
Unload driver	<code>rmmmod wilc1000</code>

## 4.6 Example Scenario with EAP server



```
$ vi /etc/wilc_hostapd_XXX.conf
```

```
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ssid=wilc1000_D4
dtim_period=2
beacon_int=100
channel=7
hw_mode=g
max_num_sta=8
ap_max_inactivity=300
ieee8021x=1
own_ip_addr=192.168.123.1
auth_server_addr=192.168.123.153
```

```
auth_server_port=1812
auth_server_shared_secret=testing123
wpa=2
wpa_key_mgmt=WPA-EAP
wpa_pairwise=TKIP CCMP
rsn_pairwise=TKIP CCMP
```

```
$ hostapd /etc/wilc_hostapd_xxx.conf -B
$ brctl addbr br0
$ brctl addif br0 wlan0
$ brctl addif br0 eth0
$ ifconfig wlan0 0.0.0.0
$ ifconfig eth0 0.0.0.0 up
$ ifconfig br0 192.168.123.1 up
$ mkdir -p /var/lib/dhcp/
$ touch /var/lib/dhcp/dhcpd.leases
$ dhcpd br0
```

## 5 How to Work Peer to Peer(P2P)

### 5.1 Write p2p\_supPLICANT.conf

For P2P operation, one more configuration file should be prepared for p2p0 interface. The file name can be renamed by user. If there isn't a specific reason, place the file in "/etc" folder.

```
$ vi /etc/wilc_p2p_supPLICANT.conf
```

Ex)

```
ctrl_interface=/var/run/p2p_supPLICANT
device_name=wilc1000_p2p
device_type=1-0050F204-1
update_config=1
config_methods=virtual_push_button physical_display keypad
```

```
$ vi /etc/wilc_wpa_supPLICANT.conf
```

Ex)

```
ctrl_interface=/var/run/wpa_supPLICANT
update_config=1
```

### 5.2 Running as P2P mode

Start WPA SupPLICANT service. Execute wpa\_supPLICANT with the following command on the terminal.

```
# wpa_supPLICANT -B -ip2p0 -Dnl80211 -c /etc/wilc_p2p_supPLICANT.conf -N -i wlan0 -D
nl80211 -c /etc/wilc_wpa_supPLICANT.conf
```

### 5.3 Allocate IP

This chapter is almost same with chapter 4.3. But only difference is that p2p0 interface is used instead of wlan0 interface.

- Set IP as gateway.

```
$ ifconfig p2p0 192.168.123.1
```

**TIPS**

Gateway IP is defined in the dhcpd.conf.

- Write the dhcpd.conf like the following.

– /etc/dhcpd.conf

```
ddns-update-style none;
default-lease-time 600;
max-lease-time 7200;

option subnet-mask 255.255.255.0;
option domain-name-servers 168.126.63.1, 164.124.101.2; # DNS Server IP
option domain-name "sample.example"; # domain name

subnet 192.168.123.0 netmask 255.255.255.0 {
    range 192.168.123.100 192.168.123.110; # range ip
    option broadcast-address 192.168.123.255;
    option routers 192.168.123.1; # gateway ip
}
log-facility local7;
```

**TIPS**

Each value must be modified to fit the test environment.

**WARNING**

Do not copy and paste the examples from this document to prevent to convert encoding. The files should be saved as UNIX style.

- Start the DHCP Server

```
$ mkdir -p /var/lib/dhcp/
$ touch /var/lib/dhcp/dhcpd.leases
$ dhcpd p2p0
```

**TIPS**

Otherly, user can execute DHCP server after modifying “/etc/init.d/S80dhcp-server” script.

## 5.4 Connect with Peer

1. Basic P2P commands via wpa\_cli

- Start P2P device discovery

```
# wpa_cli -p/var/run/p2p_supPLICANT p2p_find
```

→ The peers should be in P2P discovery mode at that time.

- Stop ongoing P2P device discovery

```
# wpa_cli -p/var/run/p2p_supplicant p2p_stop_find
```

→ Stop the discovery mode to go on next step.

- List discovered peers

```
# wpa_cli -p/var/run/p2p_supplicant p2p_peers
```

→ The MAC address of discovered peers will be displayed. The MAC will be used in next step.

- Connect

```
# wpa_cli -p/var/run/p2p_supplicant p2p_connect <MAC_ADDR> pbc
```

→ The peer device will get the notification that is for accept this connection.

## 5.5 Disconnect with Peer and Release

```
$ wpa_cli -p/var/run/p2p_supplicant disconnect
$ wpa_cli -p/var/run/p2p_supplicant remove_network 0
$ start-stop-daemon -K -x /usr/sbin/dhccpd
$ killall wpa_supplicant
```

## 6 How to Work Concurrent Mode (Both Station and AP)

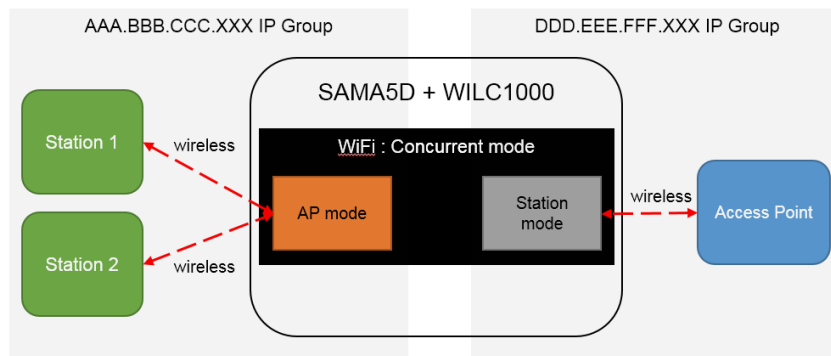
The ver12 or later of ATWILC1000 driver is supported the station/AP concurrency.

The relevant ATWILC1000 driver and firmware is available with below link:

- <https://github.com/linux4sc/wireless-driver/archive/v12.zip>
- <https://github.com/linux4sc/wireless-firmware/archive/v12.zip>

There are two interfaces created by ATWILC1000 driver: *wlan0* & *p2p0*.

For station/AP concurrency, one interface (e.g. *wlan0*) is used as station, and the other interface (e.g. *p2p0*) is used as AP.



**WARNING** The concurrent operation supports only single channel i.e. the station and AP mode should operate on the same channel.

### 6.1 Start Station mode in Concurrency

As the description of chapter 3, edit the *wpa\_supplicant.conf* file and connect to Given AP. There is no difference whether concurrent mode or not.

### 6.2 Start Access Point mode in Concurrency

Basic AP operation is same with chapter 4, but user should keep in followings:

- The *p2p0* interface is used instead of *wlan0* interface.
- The channel should be same with AP which is connected from *wilc1000* station.

#### 6.2.1 Check frequency of earlier connected AP during station mode

The connected AP's channel can be found with following command:

```
$ wpa_cli scan_results | grep User_AP
```

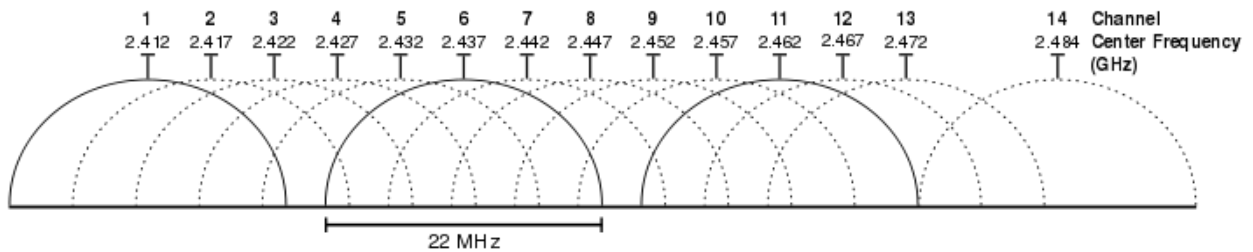
```
bssid / frequency / signal level / flags / ssid  
64:e5:99:a4:e1:8d 2462 -48 [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS] User_AP
```

Or

```
$ iw wlan0 link
```

```
Connected to 64:e5:99:a4:e1:8d (on wlan0)  
SSID: User_AP  
freq: 2462  
signal: -48 dBm  
tx bitrate: 72.0 MBit/s
```

The channel number corresponding to the frequency is shown below:



In case of above example, 2462 MHz means that operating channel of connected AP is 11.

## 6.2.2 Write hostapd.conf

Refer to the chapter 4.1 but modify the interface and channel field.

- The interface name is p2p0.
- The channel number is the value noted in chapter 6.2.1.

```
$ vi /etc/wilc_hostapd_open.conf
```

```
interface=p2p0  
...  
channel=11
```

## 6.2.3 Start Hostapd service

Refer to the chapter 4.2.

## 6.2.4 Allocate IP

Refer to the chapter 4.3 but the change the interface to p2p0 from wlan0.



- Set IP as gateway.

```
$ ifconfig p2p0 192.168.123.1
```

- Write the dhcpd.conf
- Start the DHCP Server

```
$ mkdir -p /var/lib/dhcp/
$ touch /var/lib/dhcp/dhcpd.leases
$ dhcpd p2p0
```

## 6.3 Stop Concurrent mode

Refer to the chapter 3.6 and chapter 4.4.

- Stop dhcpd server

```
$ start-stop-daemon -K -x /usr/sbin/dhcpd
```

- Stop hostapd

```
$ killall hostapd
```

- Disconnect with AP

```
$ wpa_cli -p/var/run/wpa_supplicant disconnect
$ wpa_cli -p/var/run/wpa_supplicant remove_network 0
$ killall wpa_supplicant
```

- Stop dhcpd client

```
start-stop-daemon -K -x /usr/sbin/dhcpd
```

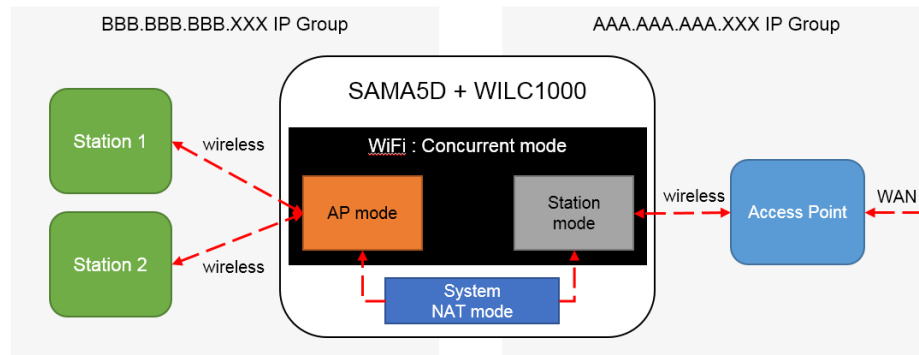
## 6.4 Internet Sharing Example on Concurrency

Establishing the NAT (Network Address Translation) is for connecting two networks, *wlan0* (external network by station mode) and *p2p0* (internal network by AP mode), in concurrency mode.

This section describes only the basic NAT used for interconnect two IP networks of *wlan0* and *p2p0* which have incompatible addressing.

- Demanded package.
  - Kernel driver : Netfilter
  - User space application : iptables

참고 Link : <https://github.com/linux4sc/wireless-driver/wiki/NAT#2-kernel-configurations>



Load driver	<code>insmod wilc1000.ko</code>
Execute wpa_supplicant	<code>wpa_supplicant -B -Dnl80211 -iwlan0 -c/etc/wilc_wpa_supplicant.conf</code>
	<code>wpa_cli -p/var/run/wpa_supplicant ap_scan 1</code> <code>wpa_cli -p/var/run/wpa_supplicant add_network</code> <code>wpa_cli -p/var/run/wpa_supplicant set_network 0 ssid ""User_AP""</code> <code>wpa_cli -p/var/run/wpa_supplicant set_network 0 key_mgmt NONE</code>
Connect with User_AP1	<code>wpa_cli -p/var/run/wpa_supplicant select_network 0</code>
Check used channel	<code>wpa_cli scan_results   grep User_AP</code>
Modify hostapd.conf	<code>vi /etc/wilc_concurrent_hostapd_open.conf</code>
Execute hostapd	<code>hostapd -B /etc/wilc_concurrent_hostapd_open.conf</code>
Set AP's IP	<code>ifconfig p2p0 192.168.123.1</code>
Execute DHCP server (optional)	<code>mkdir -p /var/lib/dhcp/</code> <code>touch /var/lib/dhcp/dhcpd.leases</code> <code>dhcpd p2p0</code>
Configure wire Ethernet card	<code>ifconfig p2p0 192.168.123.1</code>
Execute NAT(iptables)	<code>echo 1 &gt; /proc/sys/net/ipv4/ip_forward</code> <code>iptables -A FORWARD -i wlan0 -o p2p0 -j ACCEPT</code> <code>iptables -A FORWARD -o wlan0 -i p2p0 -j ACCEPT</code> <code>iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE</code>
Clients can be connected with wilc1000 SoftAP	Is doing
Terminate NAT (iptables)	<code>iptables -F</code> or <code>iptables -D FORWARD 1 &amp;&amp; iptables -D FORWARD 2</code> <code>echo 0 &gt; /proc/sys/net/ipv4/ip_forward</code>
Terminate DHCP server	<code>start-stop-daemon -K -x /usr/sbin/dhcpd</code>
Terminate hostapd	<code>killall hostapd</code>

Disconnect with User_AP	wpa_cli -p/var/run/wpa_supplicant disconnect
Terminate wpa_supplicant	killall wpa_supplicant
Unload driver	rmmmod wilc1000

## 1. How to add NAT module in Kernel

The NAT is a part of Netfilter which is the packet filtering framework inside the Linux Kernel.

The below kernel configuration shows how to include the Netfilter (NAT) based on the kernel version 3.10.0 of sama5d4\_xplained.

```
[*] Networking support --->
    Networking options --->
...
[*] Network packet filtering framework (Netfilter) ---->
...
    Core Netfilter Configuration --->
        <*> Netfilter connection tracking support
        ...
        <*> Netfilter Xtables support (required for ip_tables)
    IP: Netfilter Configuration --->
        <*> IPv4 connection tracking support (required for NAT)
            [*] proc/sysctl compatibility with old connection tracking
        <*> IP tables support (required for filtering/masq/NAT)
        ...
        <*> Packet filtering
        ...
        <*> IPv4 NAT
            <*> MASQUERADE target support
            <*> NETMAP target support
            <*> REDIRECT target support
```

## 2. How to add iptables service using buildroot package

The iptables is a program used to configure and manage the kernels netfilter modules.

Buildroot has the menu for including iptables application into the rootfs.

```
$ make menuconfig
```

Select iptables menu in:

```
Target packages --->
    Networking applications --->
        [*] iptables
```

Rebuild buildroot

```
$ make
```

## 7 Benchmark using iPerf

iPerf is a tool for active measurements of the maximum achievable bandwidth on IP network. The network link is delimited by two hosts running iPerf. One host must be set as client, the other one as server.

- Server side (receiver): downlink mode

```
#iperf -s
```

- Client side (sender): uplink mode

```
#iperf -c <server address>
```



### INFO

To get more information about iPerf, visit <https://iperf.fr/> site.

### 7.1 Testing downlink

To measure the ATWILC1000's downlink, run the ATWILC1000 in server mode and then run Android as client mode.



### RESULT

The following result is output after 10 seconds.

- ATWILC1000 output

```
#iperf -s
-----
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 4] local 192.168.43.121 port 5001 connected with 192.168.43.1 port 43232
[ ID] Interval      Transfer    Bandwidth
[ 4] 0.0-10.2 sec  25.4 MBytes  21.0 Mbits/sec
```

- Output on Android device

```
iPerf
OFF Interface: wlan0 ipaddr: 192.168.43.1
Interface: rmnet0 ipaddr: 223.44.168.174
Interface: rmnet1 ipaddr: 10.94.148.54
SHV-E330S (MSM8974) [ARM]
Wi-Fi network: Unknown SHV-E330S (MSM8974)
-c 192.168.43.121
-----
Client connecting to 192.168.43.121, TCP port 5001
TCP window size: 1.00 MByte (default)
-----
[ 3] local 192.168.43.1 port 43232 connected with
192.168.43.121 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.1 sec  25.4 MBytes  21.1 Mbits/sec
```

## 7.2 Testing uplink

To measure ATWILC1000's uplink, run Android in server mode and then run ATWILC1000 as client mode.



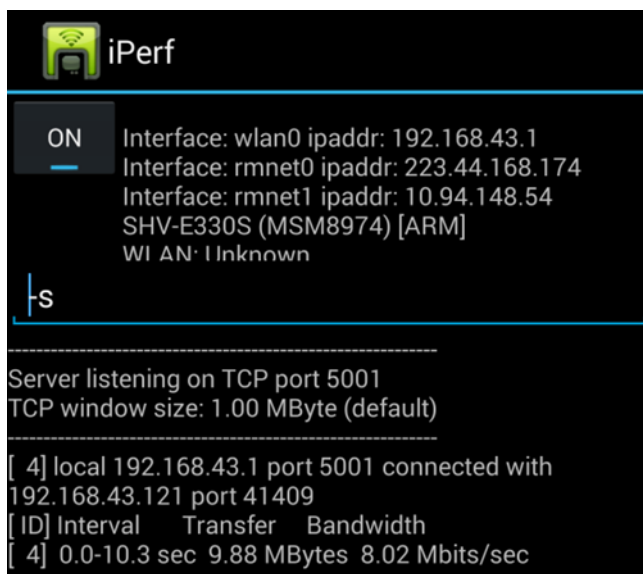
### RESULT

The following result is output after 10 seconds.

#### – ATWILC1000 output

```
#iperf -c 192.168.43.1
-----
Client connecting to 192.168.43.1, TCP port 5001
TCP window size: 20.7 KByte (default)
-----
[ 3] local 192.168.43.121 port 41409 connected with 192.168.43.1 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3]  0.0-10.1 sec  9.88 MBytes  8.23 Mbits/sec
```

#### – Output on Android device



The screenshot shows the iPerf application interface on an Android device. At the top, there is a green Wi-Fi icon and the text 'iPerf'. Below this, a toggle switch is set to 'ON'. The interface displays the following information: 'Interface: wlan0 ipaddr: 192.168.43.1', 'Interface: rmnet0 ipaddr: 223.44.168.174', 'Interface: rmnet1 ipaddr: 10.94.148.54', 'SHV-E330S (MSM8974) [ARM]', and 'Wi-Fi AN: Unknown'. Below the interface information, there is a vertical line with a small 's' at the bottom. The terminal output shows the server listening on TCP port 5001 with a TCP window size of 1.00 MByte (default). It then reports a connection from local 192.168.43.1 port 5001 to 192.168.43.121 port 41409. The final output shows the bandwidth test results: [ 4] 0.0-10.3 sec 9.88 MBytes 8.02 Mbits/sec.

```
iPerf
ON
Interface: wlan0 ipaddr: 192.168.43.1
Interface: rmnet0 ipaddr: 223.44.168.174
Interface: rmnet1 ipaddr: 10.94.148.54
SHV-E330S (MSM8974) [ARM]
Wi-Fi AN: Unknown

-----
Server listening on TCP port 5001
TCP window size: 1.00 MByte (default)
-----
[ 4] local 192.168.43.1 port 5001 connected with
192.168.43.121 port 41409
[ ID] Interval      Transfer    Bandwidth
[ 4]  0.0-10.3 sec  9.88 MBytes  8.02 Mbits/sec
```

## 8 How to use Dynamic Log Message

ATWILC1000 can use dynamic log message using debugfs.

### 8.1 Debugfs Mount

When ATWILC1000 is mounted, a 'wilc\_wifi' folder is created in the '/sys/kernel/debug/' folder.

Debugfs is automatically mounted on the system according to the setting of the target board. Otherwise, input the command with the following command.

```
# mount -t debugfs nodev /sys/kernel/debug/
```



#### RESULT

If mounted, you can find a file '/sys/kernel/debug/wilc\_wifi/wilc\_debug\_level'.

### 8.2 Change Debug Level

Default debug level is error. If you want to display the log message, change the debug level.

- Enable full log

```
# echo 1 > /sys/kernel/debug/wilc_wifi/wilc_debug_level
```

- Disable full log except error log

```
# echo 0 > /sys/kernel/debug/wilc_wifi/wilc_debug_level
```

## 9 Revision History

Doc Rev.	Date	Comments
XXXXXX	12/2015	Initial document release.



**Atmel Corporation** 1600 Technology Drive, San Jose, CA 95110 USA T: (+1)(408) 441.0311 F: (+1)(408) 436.4200 | [www.atmel.com](http://www.atmel.com)

© 2015 Atmel Corporation. / Rev.: Atmel-xxxxx-ATWILC1000\_Usage\_Guide\_12/2015.

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, SAM-BA®, and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries. ARM®, ARM Connected® logo, and others are the registered trademarks or trademarks of ARM Ltd. Windows® is a registered trademark of Microsoft Corporation in U.S. and or other countries. Other terms and product names may be trademarks of others.

**DISCLAIMER:** The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

**SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER:** Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death ("Safety-Critical Applications") without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.