ATWILC3000 Programming Guide for Android and Linux Users

AN-XXXX

Prerequisites

- **Hardware Prerequisites**
  - SAMA5D4EK Android board
  - SAMA5D3 Xplained Linux board
  - Atmel WILC3000 evaluation board (NM73000C0N_0 REV 0)
  - Micro USB Cable (TypeA / MicroB)
  - USB to Serial Adaptor (for DBGU port)

- **Build Prerequisites**
  - Linux Host PC
  - Android Software Package
  - Linux Buildroot Software Package
  - ATWILC3000 driver and firmware

Introduction

The ATWILC3000 is a single chip IEEE 802.11 b/g/n RF, baseband, MAC, Bluetooth 4.0 and FM receiver optimized for low-power mobile applications. The ATWILC3000 utilizes highly optimized 802.11 – Bluetooth coexistence protocols. It provides multiple peripheral interfaces including UART, SPI, I2C and SDIO.

This programming guide describes how to integrate the Atmel WILC3000 evaluation board via SDIO for Wi-Fi and USART for Bluetooth in SAMA5D4EK Android and SAMA5D3 Xplained Linux platforms. In addition, this programming guide introduces wireless tools to run the ATWILC3000 in the Linux platform. The following links also are available to get more information on ATWILC3000 wireless driver and firmware, Android SDK, Linux kernel, SAMA5D4EK and SAMA5D3 Xplained boards.

- Atmel SAMA5D4EK: http://www.atmel.com/tools/sama5d4-ek.aspx.
- SAMA5D3 Xplained board: http://www.atmel.com/tools/atsama5d3-xpld.aspx
- Linux4sam: http://www.at91.com/linux4sam/bin/view/Linux4SAM/
- Android4sam: http://www.at91.com/android4sam/bin/view/Android4SAM/
- ATWILC3000 official GitHub: https://github.com/atwilc3000

Table of Contents

1. Bring up SAMA5D3 Xplained Board with ATWILC3000

This section introduces how to bring up the SAMA5D3 Xplained Linux platform with ATWILC3000. For more information on how to get Linux software package, build image and flash the target board, refer to the ATWILC3000 Getting Started Guide [1].

1.1 Download SAMA5D3 Xplained Buildroot sources

Download the Buildroot-at91 software and build the image.

```
$ git clone git://github.com/linux4sam/buildroot-at91.git
$ cd buildroot-at91
$ git checkout origin/buildroot-2013.11-at91 -b buildroot-2013.11-at91
$ make sama5d3_xplained_defconfig
$ make
```

The Linux kernel sources are also downloaded while compiling the Buildroot at the /buildroot-at91/output/build/linux-83a9eb4b2f16d9b388daa473a954fb2a563a7ccb for instance. This programming guide describes how to port the ATWILC3000 kernel driver in the Linux kernel downloaded from the Buildroot.

1.2 Customizing Buildroot

Customizing Buildroot is required to bring up the ATWILC3000. Add easily packages in the rootfs by issuing the following command.

```
$ make menuconfig
```

Then, add the packages described in the following, for example, go to Target package → Networking applications and add bluez-utils with subordinates.

- Target package → Networking applications → bluez-utils
- Target package → Networking applications → dhcp (ISC)
- Target package → Networking applications → dhcpcd
- Target package → Networking applications → hostapd

1.3 Patch the kernel

This section introduces how to make rootfs image including ATWILC3000 kernel module. Find the latest ATWILC3000 driver source by issuing the following command.

```
$ git clone https://github.com/atwilc3000/driver.git
$ git clone https://github.com/atwilc3000/firmware.git
```

The patch files are also available to download from the Atmel SmartConnect ATWILC3000 GitHub [2].The kernel source is available in the **Buildroot-at91/output/build/linux-xxxxx**. Create the **atmel** directory in the **linux-xxxxx/drivers/net/wireless** if not available.

```
$ mkdir atmel
```

Copy ATWILC3000 driver downloaded from the GitHub to the **atmel** directory.

```
$ git clone https://github.com/atwilc3000/driver.git
$ cd driver
$ cp Kconfig Makefile -r wilc3000/ \
        ~/buildroot-at91/output/build/linux-xxxxx/drivers/net/wireless/atmel
```

Then, add the followings in the **Kconfig** and **Makefile** at **/drivers/net/wireless** to add the ATWILC3000 driver into the Kbuild system.

- /drivers/net/wireless/Kconfig

```
source "drivers/net/wireless/atmel/Kconfig"
```

- /drivers/net/wireless/Makefile

```
obj-$(CONFIG_ATMEL_SMARTCONNECT) += atmel/
```

Customize the kernel by issuing the following command and add ATWILC3000 driver to support Atmel wireless devices.

```
$ make ARCH=arm menuconfig
```

In the menuconfig, go to **"Device driver → Network device support → Wireless LAN → Atmel SmartConnect Wireless cards Driver"** and set the ATWILC3000 driver as module or built-in. Make sure that the SDIO interface is selected for WLAN. The ATWILC3000 device will be connected to the SAMA5D3 Xplained board via SDIO for WLAN.

Make sure that ATWILC3000 driver depends on the CFG80211 and BT package in the Linux system. Hence, check if the CFG80211 and BT is enabled or not. If disabled, add CFG80211 and BT package in the menuconfig.

For CFG80211, add "**Networking support → Wireless → Cfg80211 – wireless configuration API**".

For BT, add "**Networking support → Bluetooth subsystem support**" with subordinates. Make sure if the "**Bluetooth subsystem support → Bluetooth device drivers → HCI UART driver**" is selected with subordinates or not.

1.4 Build new rootfs image

This section describes how to make new rootfs image including the ATWILC3000 kernel module after patching the kernel with ATWILC3000 driver. The ATWILC3000 firmware should be placed in the directory, **buildroot-at91/output/target/lib/firmware** before building new rootfs image. Create **firmware** directory in the buildroot-at91/output/target/lib if not available. Then, copy the firmware downloaded from the GitHub in the firmware directory. To build new rootfs with ATWILC3000 driver in the buildroot source package, remove the following outputs.

- buildroot-at91/output/build/linux-xxxxx/.stamp_built
- buildroot-at91/output/build/linux-xxxxx/.stamp_target_installed
- buildroot-at91/output/build/linux-xxxxx/.stamp_images_installed

Then, build the rootfs by issuing the following command in the buildroot-at91 directory.

```
$ make
```

Check if the **atmel/wilc3000.ko** is generated in the following path:

- /buildroot-at91/output/target/lib/modules/3.10.0/kernel/drivers/net/wireless

Then, flash new image to the target board. Refer to the Getting Started Guide to flash the SAMA5D3 Xplained board.

2. Linux Wireless Tools

This section introduces how to run the AWILC3000 with wireless tools in the Linux OS. The wireless tools are used to scan networks, connect to desired network, and show the link status etc. The **iw** tool is a new nl80211

based CLI (Command Line Interface) program for wireless devices. The **iwconfig** tool is deprecated, which uses Wireless Extension interface and so **iw** is strongly recommended. This section also introduces the WPA supplicant which is a user space daemon that runs in the background and acts as the backend component controlling the wireless connection. The front-end tool, **wpa_cli** is a text-based program for interacting with WPA supplicant.

2.1 iw

The **iw** command is the basic tool for wireless drivers based on the nl80211 [3]. It allows a user to scan access points, get the information on wireless devices and connect to an open network or WEP protected network. The followings are the most common uses of iw command.

iw dev: listing all network interfaces for wireless hardware

iw wlan0 info: showing information for the interface, for example, here wlan0

iw wlan0 link: showing connection status

iw wlan0 scan: scanning access points

iw wlan0 connect <ssid>: connecting to an open network with <ssid>

iw wlan0 connect <ssid> key 0:1122334455: connecting to the WEP protected network

2.2 WPA supplicant

The **iw** tool doesn't support to connect to the WPA/WPA2 protected networks. Note that WPA supplicant [4] supports open, WEP and WPA/WPA2 networks. The following example is the most common cases to start WPA supplicant in the background. Make sure the WPA supplicant should run in the background before using **wpa_cli** command. For more information on WPA supplicant command line options, refer to the WPA supplicant README [5].

```
$ wpa_supplicant –B –iwlan0 –Dnl80211 –c/etc/wpa_supplicant.conf
```

For more debugging log, start WPA supplicant daemon with debugging option enabled.

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

The WPA supplicant is configured by using a text file that lists all accepted networks and security policies including pre-shared keys. Visit here to find more information and options on the WPA supplicant configuration file. The configuration file is usually located in the **/etc/wpa_supplicant.conf**. The following example is simple WPA supplicant configuration. Create the **wpa_supplicant.conf** file at the **/etc/** If not available and add the followings.

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

The **wpa_cli** is a text-based frontend program for interacting with WPA supplicant. It is used to query current status, change configuration, and trigger events and request interactive user inputs. There are many options and commands on wpa_cli. Find the most common commands and options in below. For more information on options and commands, refer to the WPA supplicant README [5].

wpa_cli commands
status = get current WPA/EAPOL/EAP status
scan = request new BSS scan
scan_results = get latest scan results
list_networks = list configured networks
select_network <network_id> = select a network (disable others)
add_network = add a network
enable_network <network_id> = enable a network
disable_network <network_id> = disable a network
disconnect = disconnect and wait for reassociate command before connecting
reassociate = force reassociation

wpa_cli options
mode: IEEE 802.11 operation mode
0 = infrastructure (Managed) mode, default
1 = IBSS (ad-hoc, peer-to-peer)
2 = AP mode

auth_alg: list of allowed IEEE 802.11 authentication algorithms
OPEN = open system authentication (required for WPA/WPA2)
SHARED = shared key authentication (requires static WEP keys)
LEAP = LEAP/Network EAP (only used with LEAP)
If not set, automatic selection is used (open system with LEAP enabled if LEAP is allowed as one of the EAP methods)

key_mgmt: list of accepted authenticated key management protocols
WPA-PSK =WPA pre-shared key (this requires 'psk' field)
WPA-EAP = WPA using EAP authentication
IEEE8021X = IEEE 802.1X using EAP authentication and (optionally) dynamically generated WEP keys
NONE = WPA is not used. Plaintext or static WEP could be used
WPA-PSK-SHA256 = Like WPA-PSK but using stronger SHA256-based algorithms
WPA-EAP-SHA256 = Like WPA-EAP but using stronger SHA256-based algorithms
If not set, this defaults to: WPA-PSK WPA-EAP

psk: WPA 256-bit pre-shared key

proto: list of accepted protocols
WPA = WPA/IEEE 802.11i/D3.0
RSN = WPA2/IEEE 802.11i (also WPA2 can be used as an alias for RSN)
If not set, this defaults to: WPA RSN

2.3 Enable the network interface

Make sure the ATWILC3000 module is loaded in the kernel by issuing the following command.

```
$ insmod /lib/modules/3.10.0/kernel/drivers/net/wireless/Atmel/wilc3000/wilc3000.ko
```

This will enable the ATWILC3000 device and download the WLAN firmware.

```
$ ifconfig wlan0 up
```

Check if the interface is wlan0 or not before enabling the interface by issuing the command.

```
$ ifconfig -a
```

2.4 Scan neighbors

To scan neighboring networks, issue the following command. A list of networks with information on for example, SSID, WPS and signal levels will be shown when the scan is completed.

```
$ iw wlan0 scan
```

The following command shows how to scan networks with WPA supplicant. Make sure the WPA supplicant should run in the background before using **wpa_cli** command to scan networks.

```
$ wpa_cli –iwlan0 scan
```

Then, check the scan results.

```
$ wpa_cli –iwlan0 scan_results
```

2.5 Connect to an unprotected and WEP protected network

WEP (Wired Equivalent Privacy) is a security algorithm for IEEE 802.11 wireless networks. The standard 64-bit WEP key is entered as a string of 10 hexadecimal characters (0-9 and A-F), and also as 5 ASCII characters (0-9, a-z, A-Z). The 128-bit WEP protocol was introduced to tighten up security and the key is entered as a string of 26 digit key of hexadecimal characters, 13 ASCII characters. There are two methods of authentication for WEP: open system and shared key authentication.

Open system authentication: the client doesn't need to provide its credentials to the access point during authentication. Any client can authenticate with the access point and then attempt to associate. The clients should have the correct keys defined in the access point to encrypt the data frames.

Shared key authentication: the clients authenticate with the access point with WEP key. The client sends an authentication request to the access point. The access point replies with a plaintext challenge. The client encrypts the challenge text using the pre-shared WEP key, and sends it back with another authentication request. Then the access point de-crypts the response, and if it matches, the client authenticates and associates with the access point.

Connect to the network found while scanning neighboring networks. The wireless tool, **iw** can connect to open or WEP protected network. The following command shows how to connect to open access point. Assume that the SSID of AP (access point) is ATMEL_AP.

```
$ iw wlan0 connect ATMEL_AP
```

The following example shows how to connect to WEP protected AP with SSID, ATMEL_AP and passphrase, 1122334455.

```
$ iw wlan0 connect ATMEL_AP key 0:1122334455
```

To connect to open network with WPA supplicant, add network and assign an index number to a new network for the specified interface. The command, add_network will return zero as the index number if no network was connected yet. The command, set_network will use this index to configure all of things.

```
$ wpa_cli –iwlan0 add_network
$ wpa_cli –iwlan0 set_network 0 mode 0
$ wpa_cli –iwlan0 set_network 0 auth_alg OPEN
$ wpa_cli –iwlan0 set_network 0 key_mgmt NONE
$ wpa_cli –iwlan0 set_network 0 ssid `"ATMEL_AP"'
$ wpa_cli –iwlan0 enable_network 0
```

Check the link status with **wpa_cli** or **iw**.

```
$ wpa_cli -iwlan0 status
```

```
$ iw wlan0 link
```

The following shows how to connect to the access point secured with open WEP 64bit (Assume that the SSID is ATMEL_AP and WEP key is 1122334455).

```
$ wpa_cli –iwlan0 add_network
$ wpa_cli –iwlan0 set_network 0 mode 0
$ wpa_cli –iwlan0 set_network 0 auth_alg OPEN
$ wpa_cli –iwlan0 set_network 0 key_mgmt NONE
$ wpa_cli –iwlan0 set_network 0 wep_key0 1122334455
$ wpa_cli –iwlan0 set_network 0 ssid `"ATMEL_AP"'
$ wpa_cli –iwlan0 enable_network 0
```

2.6 Connect to WPA/WPA2 protected network

The WPA/WPA2 was introduced because the authentication with either open system or shared key WEP is seriously weak. The WPA was an intermediate measure to take over the WEP security, and then it has been replaced by WPA2 also.

To connect WPA network which has a SSID, ATMEL_AP and passphrase, 12345678), refer to the following commands.

```
$ wpa_cli –iwlan0 add_network
$ wpa_cli –iwlan0 set_network 0 mode 0
$ wpa_cli –iwlan0 set_network 0 auth_alg OPEN
$ wpa_cli –iwlan0 set_network 0 key_mgmt WPA-PSK
$ wpa_cli –iwlan0 set_network 0 psk `"12345678"'
$ wpa_cli –iwlan0 set_network 0 ssid `"ATMEL_AP"'
$ wpa_cli –iwlan0 enable_network 0
```

To connect WPA2 network, issue the following commands.

```
$ wpa_cli –iwlan0 add_network
$ wpa_cli –iwlan0 set_network 0 mode 0
$ wpa_cli –iwlan0 set_network 0 auth_alg OPEN
$ wpa_cli –iwlan0 set_network 0 key_mgmt WPA-PSK
$ wpa_cli –iwlan0 set_network 0 proto RSN
$ wpa_cli –iwlan0 set_network 0 psk `"12345678"'
$ wpa_cli –iwlan0 set_network 0 ssid `"ATMEL_AP"'
$ wpa_cli –iwlan0 enable_network 0
```

Then, check the link status with **wpa_cli** command.

```
$ wpa_cli -iwlan0 status
```

2.7 AP mode

This section introduces the hostapd which is an IEEE802.11 AP to provide information how to configure the Atmel 802.11 wireless devices to work in access point. The hostapd is a user space daemon for AP and authentication servers. Refer to the hostapd homepage [6] for more information on supported security features, EAP methods, and wireless cards/drivers and so on. The configuration file is required like WPA supplicant, which is usually located in **/etc/hostapd.conf**. This configuration file defines the soft AP mode settings with various options See detailed information on hostapd.conf from the README [7]. The most common options used in the hostapd.conf are listed:

- interface: AP netdevice name, for example, wlan0 in the most cases.
- driver: interface type, nl80211 used with all Linux nl80211/mac80211 drivers in the most cases.
- ctrl_interface: interface for separate control program for example, hostapd_cli.
- ssid: SSID to be used in IEEE 802.11 management frames.
- beacon_int: beacon interval in TU (1024 ms), the default is 100, range 15 to 65535.
- dtim_period: DTIM (Delivery Traffic Information Message) period (default 2), 1 means every beacon includes DTIM.
- channel: channel number. The default is 0. In this case, is automatically selected.
- hw_mode: operation mode (a = 802.11a, b = 802.11b, g = 802.11g) to specify the band. (default: b)
- max_num_sta: maximum number of stations allowed in station table (limited to 2007). The default is also 2007.
- ap_max_inactivity: station inactivity limit. If a station does not send anything in ap_max_inactivity seconds, an empty data frame is sent to it in order to verify whether it is still in range.
- leee80211n: if 1, then IEEE 802.11n is enabled.
- auth_algs: authentication algorithms.
- wpa_pairwise: encryption algorithms. TKIP for WPA, CCMP for WPA2.
- wpa_key_mgmt: key management algorithms (WPA-PSK, WPA-EAP, or both).
- WPA: bit0=WPA (WPA=1), bit1=WPA2 (WPA=2)

The following examples show how to write the hostapd.conf according to AP modes.

```
#Non-secured AP mode
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ssid=ATMEL_AP
dtim_period=2
beacon_int=100
channel=6
auth_algs=3
hw_mode=g
max_num_sta=8
ap_max_inactivity=300
```

```
#Secured AP mode with WPA
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ssid=ATMEL_AP
dtim_period=2
beacon_int=100
channel=6
auth_algs=1
hw_mode=g
max_num_sta=8
ap_max_inactivity=300
wpa=1
wpa_passphrase=12341234
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
```

```
#Secured AP mode with WPA2
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ssid=ATMEL_AP
dtim_period=2
beacon_int=100
channel=6
auth_algs=1
hw_mode=g
max_num_sta=8
ap_max_inactivity=300
wpa=2
wpa_passphrase=12341234
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
```

```
#Secured AP mode with WPA/WPA2
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ssid=ATMEL_AP
dtim_period=2
beacon_int=100
channel=6
auth_algs=1
hw_mode=g
max_num_sta=8
ap_max_inactivity=300
wpa=3
wpa_passphrase=12341234
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP TKIP
```

It's required to configure the DHCP server on the AP. For the simple demo, write the **/etc/dhcpcd.conf** like the followings.

```
interface=wlan0
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_domain"; # domain name
subnet 192.168.0.0 netmask 255.255.255.0 {
        range 192.168.0.100 192.168.0.110    # range IP
        option braodcasr-address 192.168.0.255;
        option routers 192.168.0.1;             # gateway

}

Log-facility local7;
```

Then, update the **/etc/init.d/S80dhcp-server** like the following:

```
INTERFACES="wlan0"
…
test –f /etc/dhcpd.conf || exit 0
```

To run hostapd in SAMA5D3 Xplained Linux board, issue the followings.

```
$ hostapd /etc/hostapd.conf –B
$ ifconfig wlan0 192.168.0.1
$ /etc/init.d/S80dhcp-server start
```

2.8 P2P mode

This section describes how to run P2P (Wi-Fi Direct) in Linux. The WPA supplicant also supports the P2P mode for wireless drivers, and also p2p_supplicant.conf is needed as wpa_supplicant.conf does. The following is simple P2P configuration file for simple demo. This configuration file indicates the device name is wilc3000, for instance.

```
ctrl_interface=/var/run/wpa_supplicant
device_name=wilc3000
device_type=1-0050F204-1
update_config=1
config_methods=virtual_push_button physical_display keypad
```

Then, run WPA supplicant in background.

```
$ wpa_supplicant –Dnl80211 –p2p0 –c/etc/p2p_supplicant –N –Dnl80211 \
        –c/etc/wpa_supplicant –iwlan0 &
```

Configure the IP for p2p interface. Check the p2p0 interface with **ifconfig –a** command. Make sure the driver should be inserted by issuing **insmod** command before checking the interface. Assign proper IP number specific to the test environment.

```
$ ifconfig p2p0 192.168.0.1 netmask 255.255.255.0
```

Refer to the DHCP server if required and run WPA supplicant to get the Wi-Fi direct commands. Then, issue the Wi-Fi direct commands for simple demo with specific mac address like the following commads.

```
$ wpa_cli –ip2p0
>P2p_find
>p2p_stop_find
>p2p_peers
>p2p_connect da:57:ef:c7:20:42 pbc go_intent=15
```

3. Linux Bluetooth

This section provides how to run the ATWILC3000 for Bluetooth in brief. This section also introduces the BLueZ tools. Make sure the Bluetooth firmware should be downloaded before attaching the Bluetooth device in the target platform. Start the BT firmware by the following procedures.

First add wilc3000 module by issuing the following command.

```
$ insmod /lib/modules/3.10.0/kernel/drivers/net/wireless/Atmel/wilc3000/wilc3000.ko
```

Check the interface of wireless device.

```
$ ifconfig -a
```

Then, enable the interface. Assume that the interface of ATWILC3000 device is wlan0.

```
$ ifconfig wlan0 up
```

The firmware should be located in **/lib/firmware** on your target platform. Then, it's possible to attach the ATWILC3000 to the target platform.

3.1 Attach serial device via UART HCI to BlueZ stack

The ATWILC3000 Bluetooth driver provides the UART interface right now so the following command should be issued to attach the device. Make sure the **/dev/ttyS3** exists on the target platform. The BT firmware has the baud rate, 1375000 and flow control.

```
# hciattach ttyS3 any 1500000 flow
```

Make sure the HCI interface is created.

```
# hciconfig -a
hci0:   Type: BR/EDR  Bus: UART
        BD Address: AB:89:67:45:23:01  ACL MTU: 1021:9  SCO MTU: 255:4
        DOWN
        RX bytes:574 acl:0 sco:0 events:27 errors:0
        TX bytes:411 acl:0 sco:0 commands:27 errors:0
        Features: 0xff 0xff 0xcd 0xfe 0xdb 0xff 0x7b 0x87
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH HOLD SNIFF PARK
        Link mode: SLAVE ACCEPT
```

3.2 Enable the network interface

Bring up the ATWILC3000 Bluetooth HCI interface.

```
# hciconfig hci0 up
```

3.3 Scan for networks

Scan for neighboring networks. A list of networks with information on BD_ADDR and name will be shown when the scan is completed.

```
# hcitool scan
Scanning ...
        60:6C:66:A4:29:63   D247-PC
        60:03:08:89:93:E7   damiank-mbp1
        48:D2:24:63:5B:C3   n/a.
        E0:06:E6:BE:A8:FA   APDN194
        D8:57:EF:C7:20:4D   SHV-E210K
        78:DD:08:B2:91:C9   ALEX-PC
```

3.4 Connect to network

You can use DBUS interface so that you connect to a network that was found during the scan. In this case, you have to choose the MAC address. Make sure Bluetooth daemon should first run.

```
$ bluetoothd &
```

Issue the following command to connect to the network with MAC address, for example, the MAC address is D8:57:EF:C7:20:4D in this demo.

```
$ export BT_ADAPTER=`dbus-send --system --dest=org.bluez --print-reply /
org.bluez.Manager.DefaultAdapter | tail -n 1 | sed 's/^.*"\(.*\)".*$/\1/'`

$ dbus-send --system --print-reply --dest=org.bluez $BT_ADAPTER
org.bluez.Adapter.CreatePairedDevice string: D8:57:EF:C7:20:4D
objpath:/org/bluez/agent string:NoInputNoOutput
```

The following commands list up the paired device.

```
$ dbus-send --system --print-reply --dest=org.bluez $BT_ADAPTER
org.bluez.Adapter.ListDevices
method return sender=:1.2 -> dest=:1.11 reply_serial=2
  array [
    object path "/org/bluez/1348/hci0/dev_D8_57_EF_C7_20_4D"
  ]
```

The following commands remove the paired device.

```
$ dbus-send --system --print-reply --dest=org.bluez  $BT_ADAPTER
org.bluez.Adapter.RemoveDevice objpath:$BT_ADAPTER/dev_D8_57_EF_C7_20_4D
```

4. Bring up SAMA5D4EK Android Board

This chapter describes how to bring up the SAMA5D4EK Android board with ATWILC3000. The Android platform for AT91 is maintained at the Android4sam [8]. For more information on how to get Android software package, build image and flash the target board, refer to the ATWILC3000 Getting Started Guide [1]. Also, ATWILC3000 Getting Started Guide describes how to connect the ATWILC3000 to SAMA5D4EK board with SDIO interface for WLAN and USART interface for Bluetooth. This chapter will briefly describe how to get the Android software package and build images to port the ATWILC3000 driver. See the following instructions to get the SDK and build images. This instruction is based on NAND flash boot mode. The patch files for SAMA5D4EK are provided in the Atmel SmartConnect ATWILC3000 GitHub [2].

```
$ mkdir android4sam_v4.4_rc2
$ cd android4sam_v4.4_rc2
$ repo init -u git://github.com/Android4SAM/platform_sammanifest.git -b
android4sam_v4.4_rc2
$ repo sync
```

Then, build the image with the following commands.

```
$ . build/envsetup.sh
$ lunch sama5d4-eng
$ make
$ mkubi_image -b sama5d4
```

The android images, **system_ubifs-SAMA5D4-ANDROID-4.4.2_r2.img** and **userdata_ubifs-SAMA5D4-ANDROID-4.4.2_r2.img** are generated in the Android root directory if successful. To download and build the kernel image refer to the following commands.

```
$ git clone git://github.com/Android4SAM/linux-at91.git
$ cd linux-at91
$ git checkout -b linux-at91 Android4sam_v4.4_rc2
```

Make sure root directory at the /android4sam_v4.4_rc2/out/target/product/sama5d4 should be copied in the /linux-at91 directory before building kernel image.

```
$ cp –r /android4sam_v4.4_rc2/out/target/product/sama5d4/root ../linix-at91
```

To build kernel zImage, wilc3000.ko and Device Tree binaries, issue the command. Assume that a cross compiler, arm-linux-gnueabi is installed in the Linux machine.

```
$ make mrproper
$ make ARCH=arm sama5_android_defconfig
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- zImage
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- dtbs
```

Then, flash new images generated to the target board. Refer to the ATWILC3000 Getting Started Guide [1].

4.1 Patch the Android

This section introduces how to patch the Android SDK to bring up the ATWILC3000 in the SAMA5D4EK board. The Android patch files are downloadable from the Atmel SmartConnect ATWILC3000 GitHub [2].

4.1.1 device/Atmel/sama5d4/BoardConfig.mk

The ATWILC3000 WLAN and Bluetooth are enabled in android with the following configuration in the BoardConfig.mk file.

```
BOARD_WIFI_VENDOR := atmel
ifeq ($(BOARD_WIFI_VENDOR), atmel)
        CONFIG_DRIVER_WEXT :=y
        BOARD_WPA_SUPPLICANT_DRIVER := NL80211
        WPA_SUPPLICANT_VERSION      := VER_0_8_X
        BOARD_WPA_SUPPLICANT_PRIVATE_LIB := lib_driver_cmd_nmc
        BOARD_HOSTAPD_DRIVER        := NL80211
        BOARD_HOSTAPD_PRIVATE_LIB   := lib_driver_cmd_nmc
        WIFI_FIRMWARE_LOADER              := ""
        BOARD_WLAN_DEVICE := wilc3000
        WIFI_DRIVER_FW_PATH_STA := "AUTO"
        WIFI_DRIVER_FW_PATH_AP  := "AUTO"
        WIFI_DRIVER_FW_PATH_P2P := "AUTO"
        BOARD_HAVE_BLUETOOTH :=true
        BOARD_HAVE_BLUETOOTH_ATMEL := true
        SW_BOARD_HAVE_BLUETOOTH_NAME := wilc3000
endif
```

4.1.2 device/atmel/sama5d4/device.mk

Add WLAN and Bluetooth firmware at the PRODUCT_PACKAGES.

```
PRODUCT_PACKAGES += \
                wifi_firmware_p2p_concurrency.bin \
                wifi_firmware.bin \
                wifi_firmware_ap.bin \
                bt_fw_BR921600_FC_Off.bin \
                bt_fw_BR921600_FC_13_21.bin       \
                bt_fw_BR1375000_FC_13_21.bin \
                iwlist \
                iwconfig
```

The Bluetooth feature is enabled with the following configuration in the device.mk file.

```
PRODUCT_COPY_FILES += \
        frameworks/native/data/etc/android.hardware.bluetooth.xml:system/etc/permissio
        ns/android.hardware.bluetooth.xml \
        frameworks/native/data/etc/android.hardware.bluetooth_le.xml:system/etc/permis
        sions/android.hardware.bluetooth_le.xml \
```

4.1.3 device/atmel/sama5d4/init.sama5-pda.rc

This file sets owner, group and also permission for the WLAN and Bluetooth like the following configuration.

```
insmod /system/lib/modules/wilc3000.ko
chown bluetooth net_bt_stack /dev/at_bt_pwr
#UART device
chmod 0660 /dev/ttyS2
chown bluetooth net_bt_stack /dev/ttyS2
chown bluetooth net_bt_stack /dev/at_bt_pwr
```

The WPA supplicant is also configured with the following configuration for WLAN Station, AP and P2P mode.

```
service wpa_supplicant /system/bin/logwrapper /system/bin/wpa_supplicant -dd \
        -iwlan0 -Dnl80211 -c/data/misc/wifi/wpa_supplicant.conf \
        -O/data/misc/wifi/sockets \
        -e/data/misc/wifi/entropy.bin -g@android:wpa_wlan0
        class main
        socket wpa_wlan0 dgram 660 wifi wifi
        disabled
        oneshot

service p2p_supplicant /system/bin/logwrapper /system/bin/wpa_supplicant -dd \
        -ip2p0 -Dnl80211 -c/data/misc/wifi/p2p_supplicant.conf \
        -e/data/misc/wifi/entropy.bin -N \
        -iwlan0 -Dnl80211 -c/data/misc/wifi/wpa_supplicant.conf \
        -O/data/misc/wifi/sockets \
        -g@android:wpa_wlan0
        class main
        socket wpa_wlan0 dgram 660 wifi wifi
        disabled
        oneshot
```

4.1.4 device/atmel/common/config/Android_Copy.mk

```
PREBUILD_FIRMWARE := wilc3000

ifeq ($(PREBUILD_FIRMWARE), wilc3000)
PRODUCT_COPY_FILES += \
        $(LOCAL_PATH)/wilc3000.ko:system/lib/modules/wilc3000.ko
endif
```

4.1.5 hardware/atmel/sama5dx/Android.mk

Add libbt and wlan modules in the Android.mk file.

```
modules := libagl gralloc hwcomposer audio liblights hardwareloader camera libbt wlan
```

4.1.6 hardware/atmel/sama5dx/libbt/

Patch Atmel libbt directory with the patch files provided from ATWILC3000 GitHub to build libbt_vendor.so file, which is specific to the ATWILC3000 Bluetooth controller.

4.1.7 hardware/atmel/sama5dx/wlan/

Patch Atmel wlan directory with the patch files provided from ATWILC3000 GitHub. The ATWILC3000 WLAN and Bluetooth firmware are placed in the firmware directory.

4.1.8 external/wpa_supplicant_8/

Patch WPA supplicant with the patch files provided from ATWILC3000 GitHub to run ATWILC3000 WLAN.

4.2 Patch the kernel

This section describes how to patch the kernel to bring up the ATWILC3000 with SAMA5D4EK board. Refer to the ATWILC3000 Getting Started Guide [1] to get more information on the hardware connection and how to build the images. The kernel patch files are also provided in the Atmel SmartConnect ATWILC3000 GitHub [2].

4.2.1 arch/arm/boot/dts/sama5d4.dtsi

The SAMA5D4EK supports USART DMA with adding the followings in the USART2 section.

```
usart2: serial@fc008000 {
        compatible = "atmel,at91sam9260-usart";
        reg = <0xfc008000 0x100>;
        interrupts = <29 IRQ_TYPE_LEVEL_HIGH 5>;
        atmel,use-dma-rx;
        atmel,use-dma-tx;
        dmas = <&dma1
                (AT91_XDMAC_DT_MEM_IF(0) | AT91_XDMAC_DT_PER_IF(1))
                (AT91_XDMAC_DT_PERID(16) | AT91_XDMAC_DT_DWIDTH(0x0))>,
                <&dma1
                (AT91_XDMAC_DT_MEM_IF(0) | AT91_XDMAC_DT_PER_IF(1))
                (AT91_XDMAC_DT_PERID(17) | AT91_XDMAC_DT_DWIDTH(0x0))>;
        dma-names = "tx", "rx";
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_usart2 &pinctrl_usart2_rts &pinctrl_usart2_cts>;
        status = "disabled";
};
```

The USART2 is disabled because it conflicts with ISI in ISI_VSYNC and HSYNC. However, the ATWILC3000 should use USART2 for RTS and CTS. So, the ISI should be disabled in the sama5d4ek.dts file.

4.2.2 arch/arm/boot/dts/sama5d4.dts

The USART2 should be enabled in this device tree for the ATWILC3000 Bluetooth while the ISI should be disabled.

```
isi: isi@f0008000 {
        status = "disabled"; /* For RTS and CTS, wilc3000 uses USART2 disabling ISI. */
};
```

The following statement should be included to enable USART2.

```
usart2: serial@fc008000 {
        status = "okay";
};
```

4.2.3 arch/arm/configs/sama5_android_defconfig

The followings should be added in the sama5_android_defconfig to support ATWILC3000 WLAN via SDIO and Bluetooth via USART interface.

```
CONFIG_ATMEL_SMARTCONNECT=y
CONFIG_WILC3000=m
CONFIG_WILC3000_SDIO=y
# Bluetooth device drivers
CONFIG_BT_HCIBTUSB=y
CONFIG_BT_HCIBTSDIO=y
CONFIG_BT_HCIUART=y
CONFIG_BT_HCIUART_H4=y
CONFIG_BT_HCIUART_BCSP=y
CONFIG_BT_HCIUART_LL=y
CONFIG_BT_HCIUART_3WIRE=y
```

4.2.4 drivers/net/wireless/Kconfig

The followings should be added to build ATWILC3000 driver in kernel tree.

```
source "drivers/net/wireless/atmel/Kconfig"
```

4.2.5 drivers/net/wireless/Makefile

Add the following in the Makefile.

```
obj-$(CONFIG_ATMEL_SMARTCONNECT) += atmel/
```

Note that create **atmel** directory in the wireless directory if not available. Then, copy the wilc3000 kernel driver into the **atmel** directory. The latest ATWILC3000 driver and firmware are downloadable from the Atmel SmartConnect ATWILC3000 GitHub [9].

Then, compile the kernel to make new zImage, wilc3000.ko and Device Tree binaries. In the SAMA5D4EK, the Device Tree Binary is generated at the /linux-at91/arch/arm/boot/dts/sama5d4ek.dtb. Make sure that copy root directory from Android before building new kernel images.

```
$ cp –r /android4sam_v4.4_rc2/out/target/product/sama5d4/root ../linix-at91
```

To build new zImage, issue the following command.

```
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- zImage
```

The ATWILC3000 module, wilc3000.ko file is generated.

```
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-
```

The following command generates new Device Tree Binary.

```
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- dtbs
```

The following outputs are newly generated after compile is done.

- /linux-at91/arch/arm/boot/zImage
- /linux-at91/arch/arm/boot/dts/sama5d4ek.dtb
- /linux-at91/drivers/net/wireless/Atmel/wilc3000.ko

Make sure that build new Android images by compiling Android source whenever the ATWILC3000 module, wilc3000.ko is newly generated.

Copy wilc3000.ko file in the **/android4sam_v4.4_rc2/device/Atmel/common/config/**. Then, build Android again to make Android images. The kernel zImage also should be built when new Android images are made. Copy new Android images in the kernel directory.

```
$ cp –r /android4sam_v4.4_rc2/out/target/product/sama5d4/root ../linix-at91
```

Then, build new zImage by issuing the following command.

```
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- zImage
```

Flash the target board with the following outputs. Refer to the ATWILC300 Getting Started Guide to flash the images to the target board.

- /linux-at91/arch/arm/boot/zImage
- /linux-at91/arch/arm/boot/dts/sama5d4ek.dtb
- /android4sam_v4.4_rc2/system_ubifs-SAMA5D4-ANDROID-4.4.2_r2.img
- /android4sam_v4.4_rc2/userdata_ubifs-SAMA5D4-ANDROID-4.4.2_r2.img

5. Conclusion

This Programming Guide described how to integrate the Atmel WILC3000 Combo driver in the Android platform with SAMA5D4-EK board and Linux platform with SAMA5D3 Xplained board. The wireless tools for WLAN and Bluetooth BlueZ also are introduced to run the ATWILC3000 in the Linux platform.

6. Revision history

Doc. Rev.	Date	Comments
XXXXXA	2/2015	Initial document release
	2/2015	Add wilc3000 Bluetooth
	3/2015	Update the contents

7. References

[1] ATWILC3000 docs GitHub: https://github.com/atwilc3000/driver/tree/master/docs

[2] ATWILC3000 patch GitHub: https://github.com/atwilc3000/patch

[3] NL80211: https://wireless.wiki.kernel.org/en/developers/documentation/nl80211

[4] WPA supplicant: http://w1.fi/wpa_supplicant/

[5] WPA supplicant README: http://w1.fi/cgit/hostap/plain/wpa_supplicant/README

[6] Hostapd homepage: http://w1.fi/hostapd/

[7] Hostapd configuration README: http://w1.fi/cgit/hostap/plain/hostapd/hostapd.conf

[8] Atmel Android for SAM: http://www.at91.com/android4sam/bin/view/Android4SAM/WebHome

[9] Atmel SmartConnect ATWILC3000 GitHub: https://github.com/atwilc3000

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel:(+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel:(+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel:(+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370