ATWILC1000 Programming Guide

Atmel SmartConnect

Introduction

This user guide describes how to integrate the Atmel® ATWILC1000 and work Wi-Fi on the Atmel MCU platform. Useful information of Atmel MCU is already described in Atmel for SAM website and GitHub for the relevant source code.

In Linux® case, this user guide is based on SAMA5D3 Xplained board and explains the programing in chapters 1 to 6. For the SAMA5D3 Linux platform, there a lot of useful information is already described in Atmel Linux for SAM website. The relevant source codes are maintained on GitHub. It is highly recommended the user to visit those sites. Refer to chapters 1 to 6.

- Atmel Linux For SAM Site: http://www.at91.com/linux4sam
- GitHub Linux For SAM Site: http://github.com/linux4sam

In Android case, this user guide is based on SAMA5D3-EK board and explains the programing from Chapter 7 to Chapter 10. For the SAMA5D3 Android platform, a lot of useful information is already described in Atmel Android for SAM website. The relevant source codes are maintained on GitHub. It is highly recommended the user to visit those sites.

- Atmel Android for SAM Site: http://www.at91.com/android4sam
- GitHub Android for SAM Site: http://github.com/android4sam

In non-OS case this user guide is based on SAMG53 Xplained Pro board.

Prerequisites

- Hardware Prerequisites
  - Linux
    - Atmel SAMA5D3 Xplained Evaluation Kit
    - Atmel ATWILC1000 SD Pro Card
    - USB to Serial Adaptor (for DBGU port)
  - Android
    - Atmel SAMA5D3-EK Evaluation Kit
    - Atmel ATWILC1000 SD Pro Card
    - USB to Serial Adaptor (for DBGU port)
  - Non-OS
    - Atmel SAMG53 Xplained Pro Kit
    - Atmel ATWILC1000-XPRO Wi-Fi Extension Board
  - Common
    - Micro-USB Cable (Micro-A / Micro-B)
- Build Prerequisites
  - Linux and Android
    - Linux Host PC
    - Linux and Android Software Package
  - Non-OS
    - Atmel Studio

Table of Contents

Icon Key Identifiers

INFO — Delivers contextual information about a specific topic.

TIPS — Highlights useful tips and techniques.

Info: — 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 ATWILC1000 Linux Software Package

The following site is included for running the Atmel ATWILC1000 SD Pro Card or other types.

- https://github.com/linux4sc

Software is based on the SAMA5D3 Xplained sources described in Atmel Linux-for-SAM sites.

2 How to Build Kernel With ATWILC1000 Driver

There are several versions of Linux Kernel in Linux for SAM site. Some versions are made for prebuilt image. To help the user evaluating SAMA5D3 Linux board easily, this document will utilize the following prebuilt image as a reference:

```
ftp://www.at91.com/pub/demo/linux4sam_4.3/linux4sam-poky-sama5d3_xplained-4.3.zip
```

TIPS　　　　This document does not describe general information on the `AT91Bootstrap` and `U-Boot` because it can be used without modification from prebuilt image.

It is better to check out the same branch source, to compare with the prebuilt image.

2.1 Getting Kernel Source

Get Kernel source by the following command.

```
$ git clone https://github.com/linux4sam/linux-at91.git

$ cd linux-at91

$ git checkout origin/linux-3.10-at91 -b linux-3.10-at91
```

TIPS　　　　For a reference, the branch of "linux-3.10-at91" was used and verified for AT-WILC1000 hardware.

2.2 Patching ATWILC1000 Driver Source

Add ATWILC1000 driver in Kernel source by the following command.

```
$ git clone https://github.com/linux4sc/wireless-driver.git ./drivers/net/wireless/atmel
```

TIPS　　　　Use diff-and-merge technic to see what will be changed after all.

RESULT　　In the Linux Wireless driver directory, "/driver/net/wireless" directory, ATWILC1000 Linux driver are newly added as "atmel" directory.
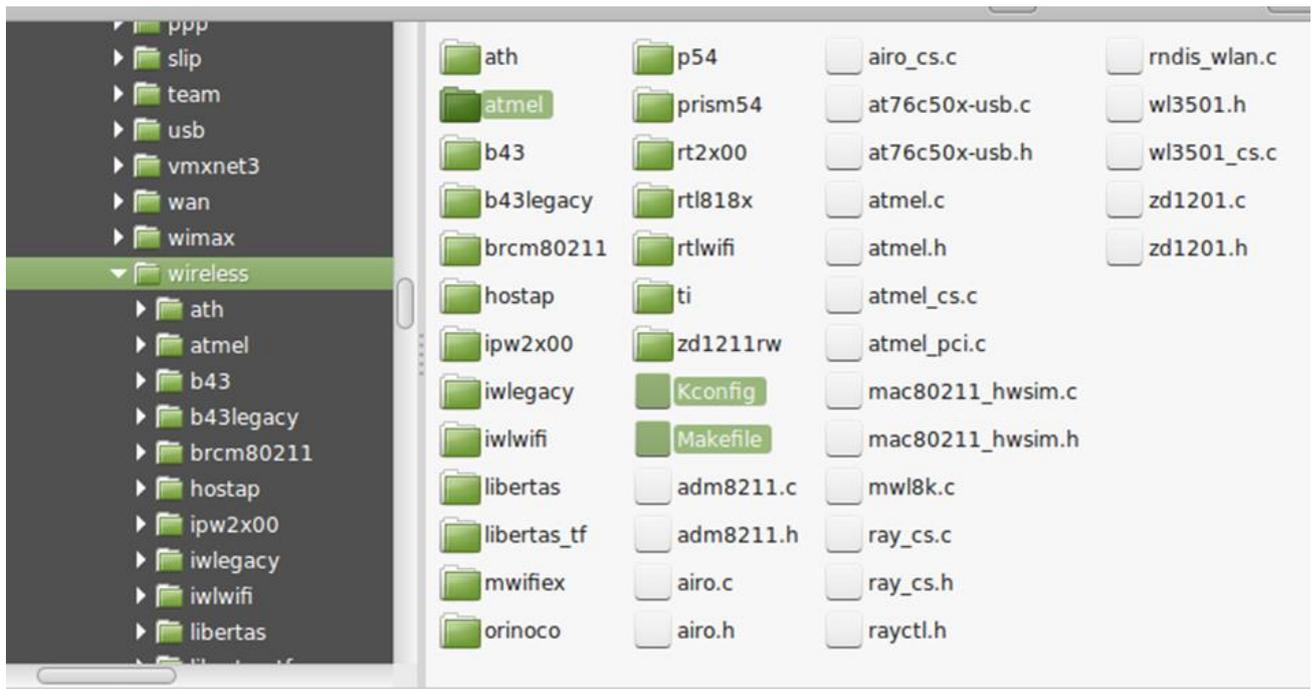
To include the ATWILC1000 driver in Kernel build, relevant Kconfig and Makefile are also changed.

- linux-at91/drivers/net/wireless/Kconfig

```
config AT76C50X_USB
… skip
source "drivers/net/wireless/atmel/Kconfig"
```

- linux-at91/drivers/net/wireless/Makefile

```
obj-$(CONFIG_AT76C50X_USB)      += at76c50x-usb.o
obj-$(CONFIG_ATMEL_SMARTCONNECT) += atmel/
obj-$(CONFIG_PRISM54)           += prism54/
```

- linux-at91/arch/arm/configs/sama5d3_xplained_defconfig

2.3 Kernel Configuration

Because users have to configure the Linux Kernel according to the platform, input the following command.

```
$ make ARCH=arm <TARGET_BOARD_DEFCONFIG>
```

INFO   SAMA5D3 Xplained board use "sama5d3_xplained_defconfig" about <TARGET_BOARD_DEFCONFIG>.

SAMA5D3 Xplained board use "`ama5d3_xplained_defconfig`" about `<TARGET_BOARD_DEFCONFIG>`.

At this step, to include ATWILC1000 Driver, you have to modify default configuration using the "menuconfig" parameter.

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

1. Common selection for ATWILC1000.

   Choose the Atmel SmartConnect menu, "Device Driver → Network device support → Wireless LAN → Atmel SmartConnect Wireless cards Driver".
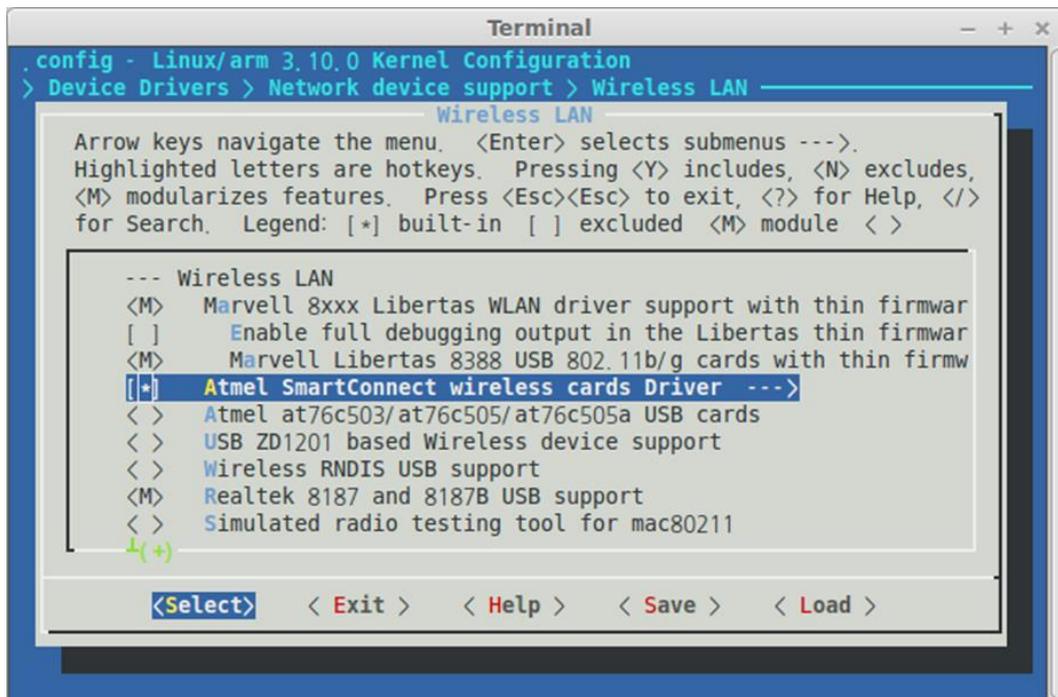
To use ATWILC1000 chipset, set to either "M" or "*" as depicted in the below.
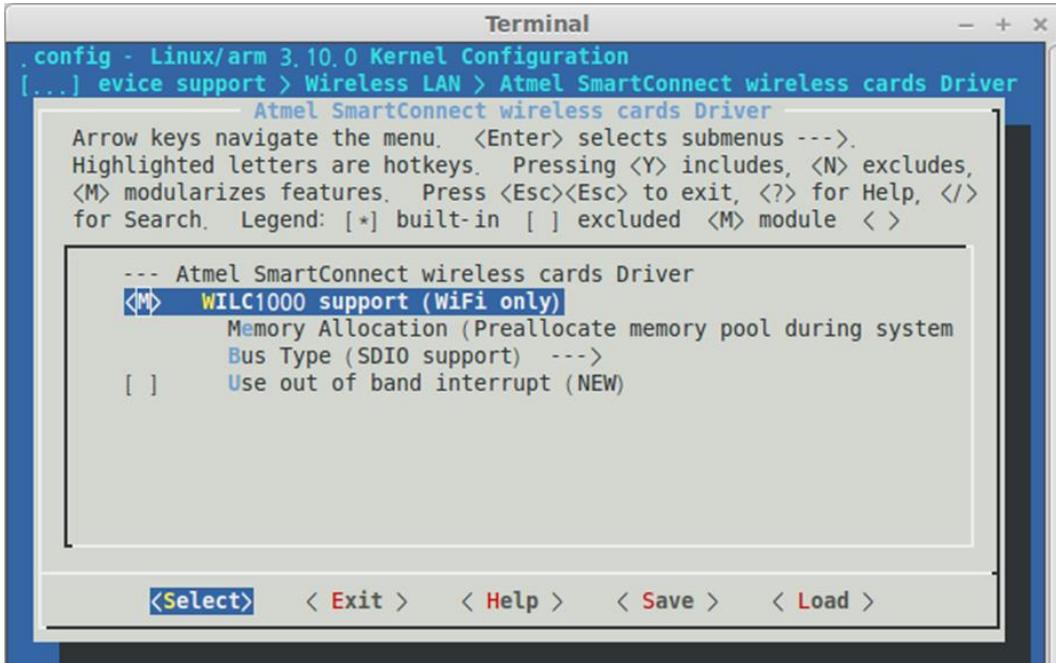
Heap area used in ATWILC1000 Driver can be chosen by user as allocation time. If system memory is shortage, choose first item.

– Allocation at boot up system
– Allocation at insmod command
– Dynamic allocation while Wi-Fi driver work
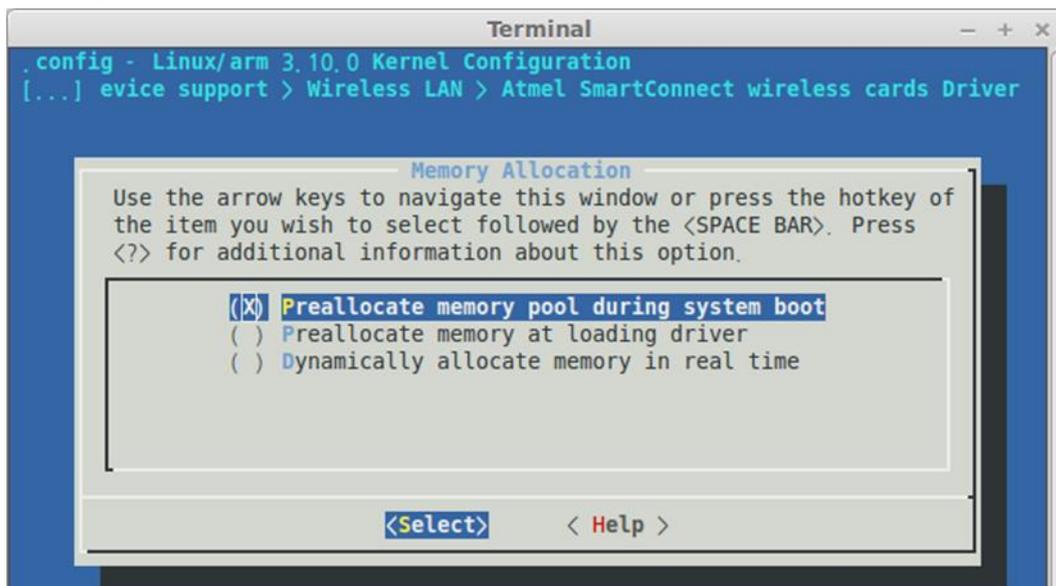
⚠ **WARNING** To communicate between wpa_supplicant of user space and driver of Kernel space nl80211 driver must be included. So check cfg80211 on networking support → wireless menu.

2. In case of using only SDIO (Secure Digital Input Output) bus.

   If WILC1000 chipset or module is connected by SDIO bus on the platform, choose "SDIO support" on bus type item. Because Atmel ATWILC1000 SD Pro Card has SDIO interface, choose this option.

3. In case of using SDIO bus with external interrupt.

   There may be some trouble about SDIO interrupt on special platform. If so, using the interface mixed SDIO bus and external IRQ pin, ATWILC1000 driver can be operate. To send interrupt signal to external pin, choose option 'Use out of band interrupt'. This option appears when SDIO is chosen.

4. In case of using SPI (Serial Peripheral Bus) bus.

If WILC1000 chipset or module is connected by SPI bus on the platform, choose "SPI support" on bus type item. Because SPI bus demands basically external IRQ pin to receive data from ATWILC1000, 'Use out of band interrupt' option is hided.

```
--- Atmel SmartConnect wireless cards Driver
<M>   WILC1000 support (WiFi only)
        Memory Allocation (Preallocate memory pool during system
        Bus Type (SPI support)  --->
```

📝 **TIPS**   Alternatively, the ".config" file in the Kernel directory can be simply editable as below.

```
CONFIG_WIRELESS=y

CONFIG_CFG80211=y

...

CONFIG_ATMEL_SMARTCONNECT=y

CONFIG_WILC1000=m

CONFIG_WILC1000_SDIO=y

CONFIG_WILC1000_PREALLOCATE_DURING_SYSTEM_BOOT=y
```

2.4   Customizing ATWILC1000 Source

1. For SDIO.

   As standard Kernel API is used, user can easy work the ATWILC1000 without the modification of source. The SDIO ATWILC1000 related protocol is implemented in the file 'wilc_sdio.c' while the Linux related SDIO API is implemented in the file 'linux_wlan_sdio.c'

   When Wi-Fi driver is loaded and unloaded on Kernel, if user want to add special commands, write it in the functions 'linux_sdio_init' and 'linux_sdio_deinit' of file 'linux_wlan_sdio.c'.

   "MAX_SPEED" value of file "linux_wlan_sdio.c" can be changed by user as platform's bus stability.

   ℹ️ **INFO**   " MAX_SPEED" value of file "linux_wlan_sdio.c" can be changed by user as plat-
                        form's bus stability.

2. For SPI.

   The SPI related protocol is implemented in the file 'wilc_spi.c' while the Linux related SPI API is implemented in the file 'linux_wlan_spi.c'. When the module is inserted, the module calls the function 'linux_spi_init' which in turn calls the function 'spi_register_driver(&wilc_bus)'.

   The wilc_bus is defined as follow in file 'linux_wlan_spi.c'.

```
struct spi_driver wilc_bus __refdata = {
      .driver = {
      .name = MODALIAS,
      },
```

Atmel

```
        .probe =  wilc_bus_probe,

        .remove = __exit_p(wilc_bus_remove),

};
```

The only thing that needs to be changed is the definition of the MODALIAS, it should be changed to match the target platform name. This value is defined at the file 'linux_wlan_common.h'.

And SPI clock can be changed to improve the communication with host processor during working. So MIN_SPEED and MAX_SPEED value has to be defined depends on the clock which platform support.

Look at it in 'linux_wlan_spi.c' file.

```
#elif defined(CUSTOMER_PLATFORM)
/*
    TODO : define Clock speed under 48M.
*/
#else
    #define MIN_SPEED 24000000
    #define MAX_SPEED 48000000
#endif
```

INFO   But if user platform handle the peripheral device by Device-tree, rightly to register ATWILC1000 SPI driver, Device ID must be added in DTS file instead of MODALIAS. SAMA5D3 Xplained board use Device-tree. Refer the following guide.

- arch/arm/boot/dts/at91-sama5d3_xplained.dts

  SPI1 channel is from 10 to 13 on J15 header of SAMA5D3 Xplained board.

```
spi1: spi@f8008000 {
        cs-gpios = <&pioC 25 0>, <0>, <0>, <&pioD 16 0>;
        status = "okay";
        wilc_spi@0 {
                compatible = "atmel,wilc_spi";
                spi-max-frequency = <6000000>;
                reg = <0>;
                status = "okay";
        };
};
```

- drivers/net/wireless/atmel/linux_wlan_spi.c

```
#ifdef CONFIG_OF
static const struct of_device_id wilc1000_of_match[] = {
        { .compatible = "atmel,wilc_spi", },
```

```
            {}
};
MODULE_DEVICE_TABLE(of, wilc1000_of_match);
#endif


struct spi_driver wilc_bus __refdata = {
            .driver = {
                        .name = MODALIAS,
#ifdef CONFIG_OF
                        .of_match_table = wilc1000_of_match,
#endif
            },
            .probe =  wilc_bus_probe,
            .remove = __exit_p(wilc_bus_remove),
};
```

3. For Interrupt.

When ATWILC1000 has SPI interface, External Pin Interrupt has to be used to transfer the data to host processor. When the module initialized it registers a GPIO (in case of SPI or SDIO with external interrupt) to act as an interrupt, the driver uses the definition GPIO_NUM, this definition should be defined in the file 'linux_wlan_common.h'.

- Check and modify it as below:
    - Define GPIO number (GPIO_NUM) which is physically mapped with ATWILC1000 in file 'linux_wlan_common.h"
    - If user has to allocate IRQ number as user platform, replace 'nic->dev_irq_num' value in function 'init_irq' of file 'linux_wlan.c'
    - If Interrupt enable and disable command isn't standard Kernel API, fix and add 'linux_wlan_enable_irq' and 'linux_wlan_disable_irq' in file 'linux_wlan.c'

INFO   SAMA5D3 Xplained board has the unused PC4 IO with SPI line on J15 socket. Because GPIO number of PC4, 8 of J15 header, is 0x44, GPIO_NUM is only defined without fixing others.

2.5   Kernel Build

The following shows how to build the Kernel.

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

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

$ make -j9 ARCH=arm  CROSS_COMPILE=arm-linux-gnueabi- dtbs
```

RESULT   The outputs are generated to the path as shown below. The 'dtb' file generated depends on which board type user has chosen.

- linux-at91/arch/arm/boot/zImage
- linux-at91/arch/arm/boot/dts/at91-sama5d3_xplained.dtb
- linux-at91/drivers/net/wireless/atmel/wilc1000/wilc1000.ko

3 How to Build Root File System

This chapter describes how to make the Linux root file system (as known as rootfs). The ATWILC1000 driver requires its firmware and some wireless utility with configuration files. They have to be included in the resulting root file system. The Atmel SAM series basically is using 'buildroot' for creating the root file system.

3.1 Getting Buildroot Source and Default Configuration

You can easily download the buildroot source as described below.

```
$ git clone https://github.com/linux4sam/buildroot-at91

$ cd buildroot-at91

$ git checkout origin/buildroot-2013.11-at91 -b buildroot-2013.11-at91
```

Input the following command.

```
$ make <TARGET_BOARD_DEFCONFIG>
```

INFO   SAMA5D3 Xplained board use "sama5d3_xplained_defconfig" for <TARGET_BOARD_DEFCONFIG>.

3.2 The Demanded Package for ATWILC1000

The default configuration may miss some of the services for the network system. Include them by the following command before building the buildroot.

```
$ make menuconfig
```

- To load ATWILC1000 firmware binaries, the below option should be included.
  Move to "System configuration → dev management → Dynamic using udev" and check.
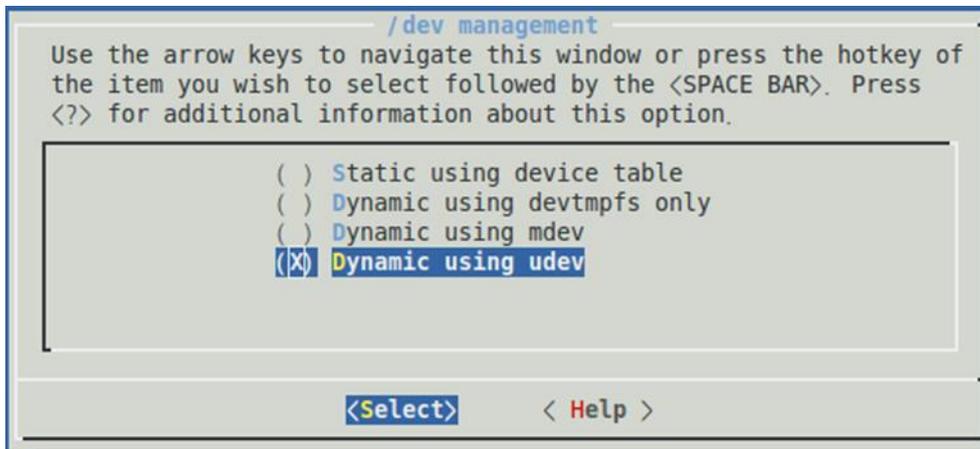
- To work Wi-Fi, the below option should be included.
  Move to "Target package → Networking applications → wpa_supplicant" and check.

- For AP mode, the below option should be included.
  Move to "Target package → Networking applications → hostapd" and check.

- To allocate IP address between server and client, the below option should be included. Move to "Target package → Networking applications → dhcp" and check.

- To communicate between WPA Supplicant with Kernel, the below option should be included. Move to "Target package → Libraries → Networking → libnl" and check.

- To connect to the secured AP, the below option should be included.
  Move to "Target packages → Libraries → Crypto" and check.

```
┌─── Crypto ────────────────────────────────────────────────────────┐
│ Arrow keys navigate the menu.  <Enter> selects submenus --->.      │
│ Highlighted letters are hotkeys.  Pressing <Y> selectes a feature, │
│ while <N> will exclude a feature.  Press <Esc><Esc> to exit, <?> for│
│ Help, </> for Search.  Legend: [*] feature is selected  [ ] feature is│
│   ┌─^(-)──────────────────────────────────────────────────────┐   │
│   │    [ ] libmhash                                            │   │
│   │    [ ] libnss                                              │   │
│   │    [ ] libsecret                                           │   │
│   │    [ ] libsha1                                             │   │
│   │    [ ] libssh2                                             │   │
│   │    [ ] nettle                                              │   │
│   │   -[*]- openssl                                            │   │
│   │    [ ]     openssl binary                                  │   │
│   │    [ ]     openssl additional engines                      │   │
│   │    [ ] polarssl                                            │   │
│   └────────────────────────────────────────────────────────────┘   │
│                                                                     │
│       <Select>    < Exit >    < Help >    < Save >    < Load >      │
└─────────────────────────────────────────────────────────────────────┘
```
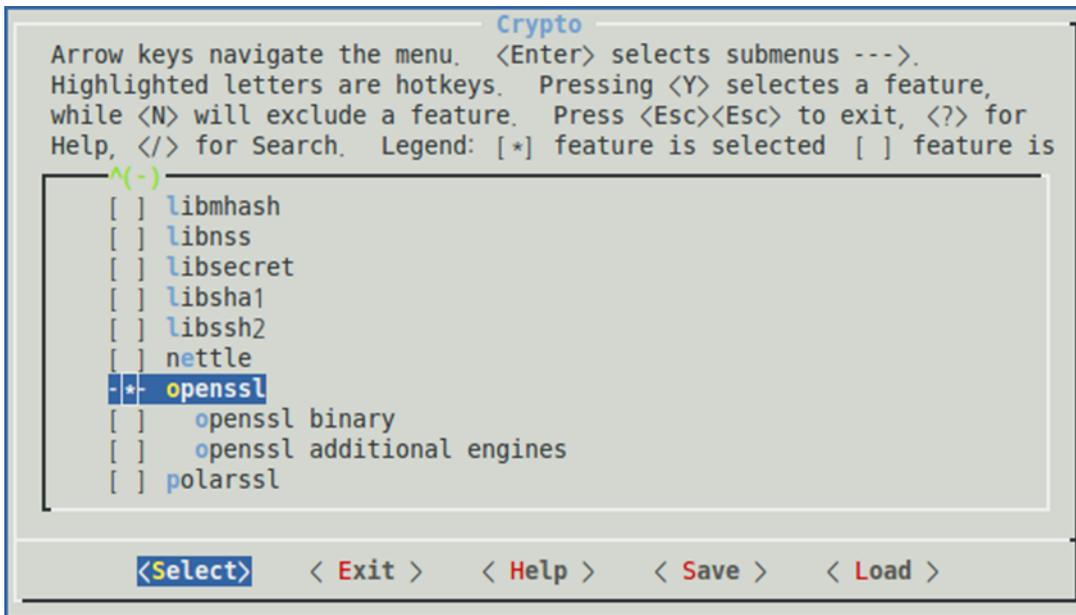
3.3 Building Buildroot

To build root file system, just make in buildroot-at91 directory.

```
$ make
```

⚠ WARNING Make sure that the host PC is connected to the internet before starting the build and no –j option. It will take long time to complete.

✔ RESULT The "rootfs.ubi" is generated in the "/output/images" folder after the building is done.
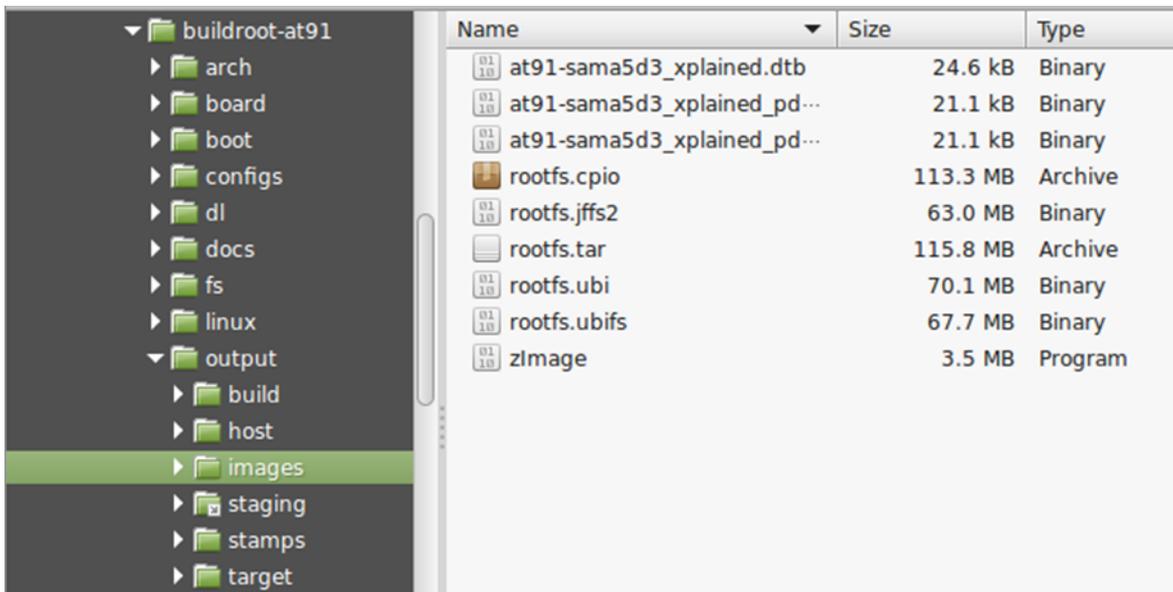
✎ TIPS If Kernel item of the buildroot's options was added, the Linux Kernel source will be downloaded during compiling the buildroot. The downloaded path is **buildroot-at91/output/build/linux-xxxx**. Then user can ignore the sequence (see Section 3.1) and can directly patch ATWILC1000 driver like sequence (see Section 3.2). If so, buildroot's binary will include the related ATWILC1000 files without adding them in file system by the user later. If you want this way, build buildroot as the following order.

- Add the ATWILC1000 driver in "buildroot-at91/output/build/linux-xxxx". See Section 2.2.
- Fix a file ".config". See Section 0.
- Copy firmware files in "buildroot-at91/output/target/lib/firmware/atmel". See Section 0.
- Add "conf" files for WPA supplicant in "buildroot-at91/output/target/etc". See Section 0, list number 1.3.
- Remove ".stamp_built", ".stamp_target_installed", and ".stamp_images_installed" in the "buildroot-at91/output/build/linux-xxxx" folder.
- Build buildroot. See Section 3.3.

4 How to Update Binary On The Target

4.1 Updating System Image

To flash the binaries to the target board, the Atmel SAM-BA® tool is required. This chapter describes how to flash the image, assuming that the SAM-BA tool is already installed in the host machine.

TIPS   Check useful resource ahead.

- http://www.at91.com/linux4sam/bin/view/Linux4SAM/SoftwareTools
- http://www.at91.com/linux4sam/bin/view/Linux4SAM/GettingStarted
- http://www.atmel.com/Images/Atmel-42051-SAM-BA-for-SAM4L_Application-Note_AT03454.pdf
- http://www.at91.com/linux4sam/bin/view/Linux4SAM/GettingStarted_xplained
- http://www.atmel.com/tools/ATSAMA5D3-XPLD.aspx?tab=documents

To make the story simpler, the user can utilize a prebuilt image.

- ftp://www.at91.com/pub/demo/linux4sam_4.3/linux4sam-poky-sama5d3_xplained-4.3.zip

1. Unzip the linux4sam-poky-sama5d3_xplained-4.3.zip.
2. Update the demo_linux_nandflash.tcl like the following.

```
set ubootFile          "u-boot-sama5d3_xplained-v2013.07-at91-r1.bin"
set kernelFile         "zImage"
set rootfsFile         "rootfs.ubi"

## board variant
```

3. Copy the following outputs to the same directory where the demo_linux_nandflash.tcl is located:
   - linux-at91/arm/boot/zImage
   - linux-at91/arch/arm/boot/dts/at91-sama5d3_xplained.dtb
   - buildroot-at91/output/images/rootfs.ubi
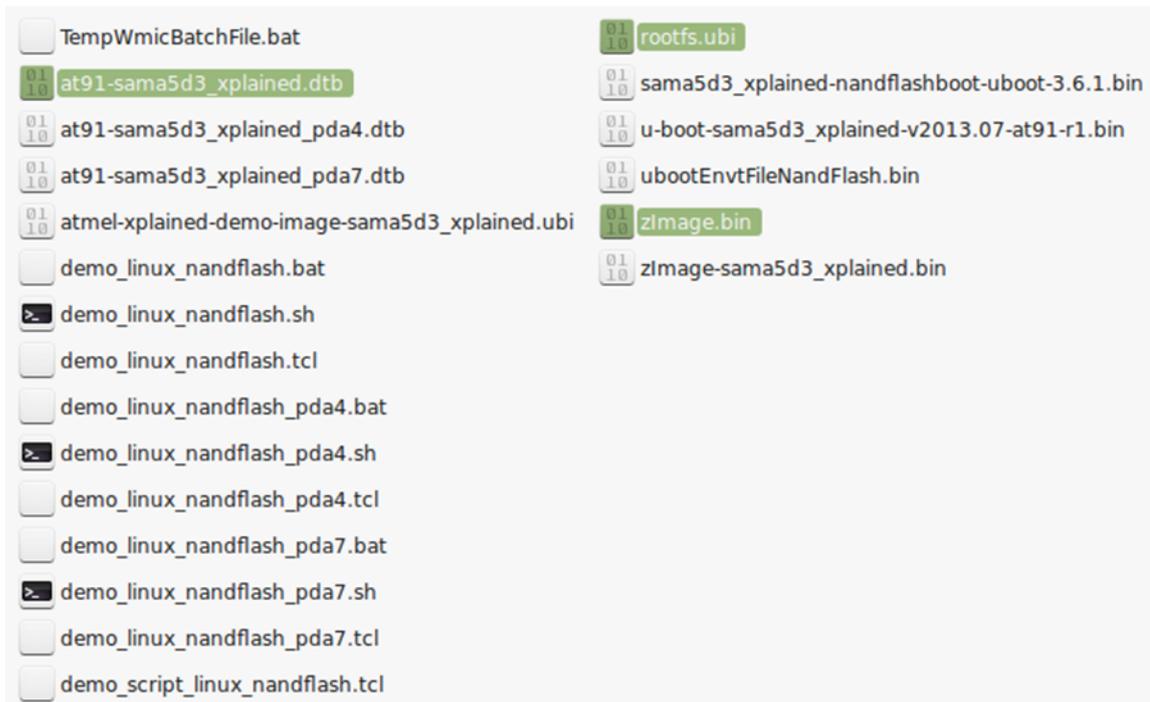
4. Open JP9 and connect to host PC via J6 port (USB). Make sure the host machine gets the USB serial port and then close a JP9.
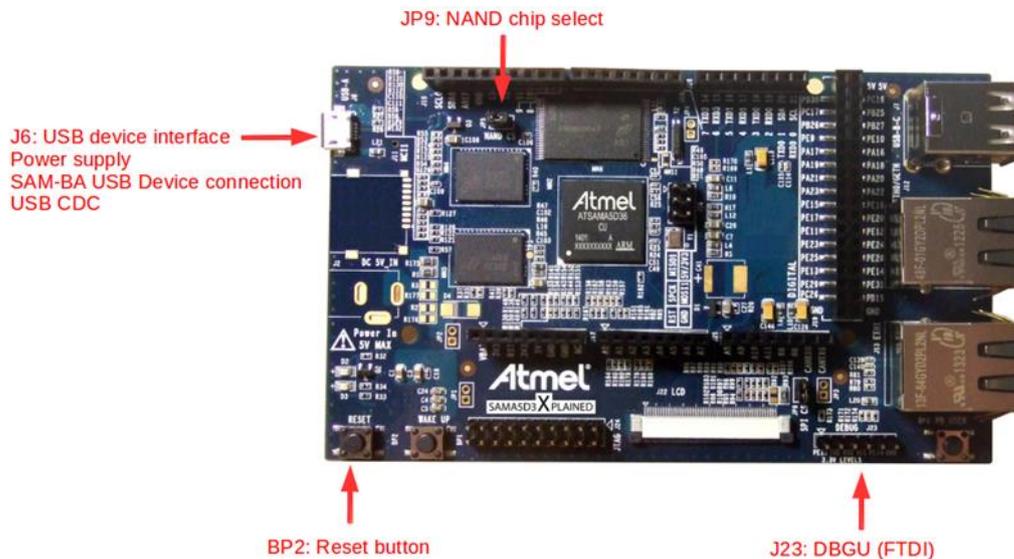
5. Execute demo_linux_nandflash.bat (Windows®) or demo_linux_nandflash.sh (Linux). The user can see the log via the J23 serial port.

Open the COM port with the following settings:

– 115200 bauds, 8-bit data, no parity, one stop bit, and no flow control

Atmel

4.2 Adding ATWILC1000 Resource

After system boot, add the related ATWILC1000 files by the following guide in file system.

1. Copy ATWILC1000 Driver module.
   Copy the following output to "/lib/modules/3.10.0/kernel/drivers/net/wireless/atmel/" or anywhere in file system:
   – linux-at91/drivers/net/wireless/atmel/wilc1000/wilc1000.ko
2. Add ATWILC1000 firmware.
   Get the firmware from https://github.com/linux4sc/wireless_firmware and copy the following output to "/lib/firmware/atmel/":
   – wilc1000_fw.bin, wilc1000_ap_fw.bin, wilc1000_p2p_fw.bin
3. Write the configuration files for WPA supplicant like the examples below:
   – "wilc_wpa_supplicant.conf" for the station mode

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

   – "wilc_hostapd.conf" for the opened AP mode

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

   – "wilc_hostapd_secure.conf" for the secured AP mode

   ```
   interface=wlan0
   driver=nl80211
   ctrl_interface=/var/run/hostapd
   ssid=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=12341234
   wpa_key_mgmt=WPA-PSK
   wpa_pairwise=TKIP CCMP
   rsn_pairwise=CCMP
   ```

5   How to Run ATWILC1000

This chapter describes how to operate the ATWILC1000 on the SAMA5D3 Xplained board.

⚠️ **WARNING**   Change log level to monitoring the detail working status. See Chapter 6.

5.1   Recognizing ATWILC1000 SDIO Card Module

SDIO card is recognized during boot up with the following line or equivalent.

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

The following shows how to load the ATWILC1000 module driver.

```
Welcome to Buildroot
buildroot login: root
# insmod /lib/modules/3.10.0/kernel/drivers/net/wireless/atmel/wilc1000/wilc1000.ko
*** WILC1000 driver VERSION=[9.4.4] REVISON=[417] FW_VER=[wilc1000_fw.bin] ***
DBG [linux_wlan_device_power: 187]linux_wlan_device_power.. (1)
DBG [linux_wlan_device_detection: 203]linux_wlan_device_detection.. (1)
DBG [linux_sdio_probe: 134]probe function
DBG [linux_sdio_probe: 145]Initializing netdev
DBG [WILC_WFI_WiphyRegister: 4446]Registering wifi device
DBG [WILC_WFI_CfgAlloc: 4393]Allocating wireless device
DBG [WILC_WFI_WiphyRegister: 4515]Successful Registering
DBG [WILC_WFI_WiphyRegister: 4446]Registering wifi device
DBG [WILC_WFI_CfgAlloc: 4393]Allocating wireless device
DBG [WILC_WFI_WiphyRegister: 4515]Successful Registering
```

5.2   Running as Station Mode

Station mode is normal Wi-Fi operation that is connecting to a given AP.

1. 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 -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 WPS supplicant, 'nl80211' option can be replaced to 'wext' option.

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

2. Connect to AP.

- 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
```

📝 **TIPS**   Change "User_AP" with the SSID of the desired AP.

- 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 '"12345678"'
# wpa_cli -p/var/run/wpa_supplicant select_network 0
```

📝 **TIPS**   Change "User_AP" and "12345678" with SSID and password of desired AP.

- 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 psk '"12345"'
#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 "12345" with SSID and ASCII (or Hex) of desired AP.

3. Run DHCP service.

If IP can be allocated from AP automatically, start the DHCP client.

```
#dhcpcd wlan0 &
```

4. 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=192.168.43.2
address=00:80:c2:b3:d7:4d
```

5.3   Running as AP Mode

1. Run Hostapd as user's configuration.

   The following command demonstrates how to execute the hostapd.

```
# hostapd /etc/wilc_hostapd.conf -B &

DBG [WILC_WFI_change_virt_intf: 3259]Changing virtual interface, enable scan
DBG [WILC_WFI_change_virt_intf: 3486]Downloading AP firmware
DBG [WILC_WFI_add_virt_intf: 4169]Adding monitor interface[cfbc6000]
DBG [WILC_WFI_add_virt_intf: 4176]Monitor interface mode: Initializing mon
interface virtual device driver
DBG [WILC_WFI_add_virt_intf: 4177]Adding monitor interface[cfbc6000]
DBG [WILC_WFI_add_virt_intf: 4181]Setting monitor flag in private structure
MAC OPEN[cfbc6000]
DBG [WILC_WFI_InitHostInt: 4562]Host[cfbc6000][cf09f140]
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)
...
```

> **TIPS** Refer to 'wilc_hostapd.conf' for the details of unencrypted AP settings. Refer and use 'wilc_hostapd_secure.conf' for WPA/WPA2 AP setting.

2. Run DHCP Server to allocate IP to client.

   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 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.100 192.168.0.110;                   # range ip
    option broadcast-address 192.168.0.255;
    option routers 192.168.0.1;                          # gateway ip
}
Log-facility local7;
```

TIPS   Each value must be modified to fit the test environment.

Update the S80dhcp-server like the following.

–   /etc/init.d/S80dhcp-server

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

Set the IP to gateway.

```
#ifconfig wlan0 192.168.0.1
```

TIPS   Gateway IP is defined in the dhcpd.conf.

Start the DHCP server.

```
#/etc/init.d/S80dhcp-server start
```

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

5.4   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.

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

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

6 How to use Dynamic Log Message

ATWILC1000 can use dynamic log message using debugfs.

6.1 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'.

6.2 Change Debug Level

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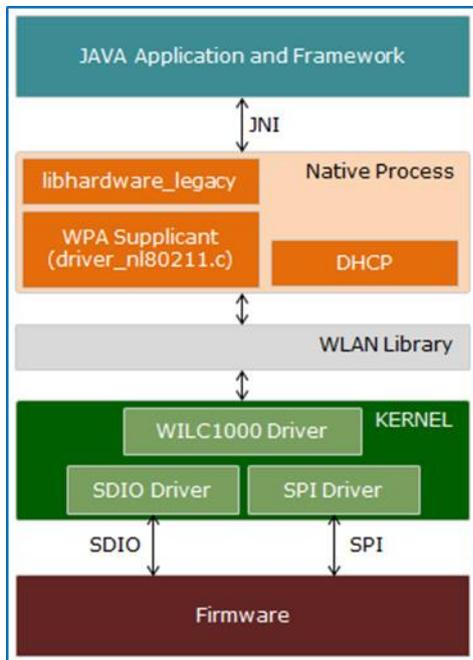
- Enable full log

```
# 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
```

7   Android Wi-Fi Architecture

The figure below shows simple stack related to ATWILC1000 on the Android platform.

Android uses a modified *wpa_supplicant* daemon for Wi-Fi support. Java® Framework communicates with wpa_supplicant using native interface wifi.c (which is part of libhardware_legacy) and its corresponding JNI implementation.

The ATWILC1000 package includes a private library of *wpa_supplicant (named lib_driver_cmd_wilc1000.a)* which has an interface for additional Android commands.

About the Kernel, refer to "Chapter 1 to 4".

7.1   Related Files

- Framework
    - JNI implementation: frameworks/base/core/jni/android_net_wifi_Wifi.cpp
- Native Process
    - libhardware_legacy: hardware/libhardware_legacy/wifi/wifi.c
    - wpa_supplicant: external/wpa_supplicant_8
- WLAN Library
    - wpa_supplicant_private_lib: hardware/atmel/wilc1000/wpa_supplicant_8_lib/
- Firmware
    - Firmware binary: hardware/atmel/wilc1000/firmware/

7.2   ATWILC1000 Android Software Packages

The following site is included for running the Atmel ATWILC1000 module board on Android 4.4.

- https://github.com/android4sc/kitkat.git

Software is based on the SAMA5D3-EK sources described in Atmel Android for SAM site.

8   Android Framework Programming

Setting up a build environment for Android is described in the AOSP website:
https://source.android.com/source/initializing.html

8.1   Get Source Code

Get the Android source by the following command.

```
$ mkdir android4sam_v4.4

$ cd android4sam_v4.4

$ repo init -u git://github.com/Android4SAM/platform_sammanifest.git

  -b android4sam_v4.4

$ repo sync
```

INFO   If there is no 'repo' utility22, install the following command.

```
$ mkdir ~/bin
$ PATH=~/bin:$PATH
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
```

TIPS   The manifest repository URL and branch could be updated. It is highly recommended to visit the Atmel Android for SAM site to check the latest status.

WARNING   The Android SDK may not be downloaded due to your internet settings. Check the connection and firewall status.

8.2   Customizing for ATWILC1000

The files below are for indicating that the device includes the Wi-Fi and Wi-Fi Direct® feature.

- android.hardware.wifi.xml
- android.hardware.wifi.direct.xml

To add hardware-specific feature on the device, the product .mk file is used as below:

```
# These are the hardware-specific features

PRODUCT_COPY_FILES += \

  frameworks/native/data/etc/handheld_core_hardware.xml:system/etc/permissions/handheld_core_hardware.xml \

  frameworks/native/data/etc/android.hardware.wifi.xml:system/etc/permissions/android.hardware.wifi.xml \

  frameworks/native/data/etc/android.hardware.wifi.direct.xml:system/etc/permissions/android.hardware.wifi.direct.xml \

  …
```

INFO   In SAMA5D3-EK, "device/atmel/sama5d3/ device.mk" is used for adding permission xml files.

1. Compile-time definitions.

   Define the following compile-time variables in "BoardConfig.mk" file to apply ATWILC1000 Wi-Fi solution into Android.

   The location of the "BoardConfig.mk" file depends on the platform.

   device / <vendor-name> / <device-name> / BoardConfig.mk

INFO   In **SAMA5D3-EK**, the "device/atmel/sama5d3/BoardConfig.mk" is used.

```
WPA_SUPPLICANT_VERSION      := VER_0_8_X

BOARD_WPA_SUPPLICANT_DRIVER := NL80211

BOARD_WPA_SUPPLICANT_PRIVATE_LIB := lib_driver_cmd_wilc1000

BOARD_HOSTAPD_DRIVER        := NL80211

BOARD_HOSTAPD_PRIVATE_LIB   := lib_driver_cmd_wilc1000

WIFI_DRIVER_MODULE_NAME     := "wilc1000"

WIFI_DRIVER_MODULE_PATH     := "/system/lib/modules/wilc1000.ko"

WIFI_DRIVER_FW_PATH_STA     := "AUTO"

WIFI_DRIVER_FW_PATH_AP      := "AUTO"

WIFI_DRIVER_FW_PATH_P2P     := "AUTO"

BOARD_WLAN_DEVICE := wilc1000
```

The meaning of each value is shown below:

   – WPA_SUPPLICANT_VERSION := VER_0_8_X

   Enable build of Android default external/wpa_supplicant_8.

   – BOARD_XXX_DRIVER := NL80211

   Enable build driver_nl80211.c in wpa_supplicant.

   – BOARD_XXX_PRIVATE_LIB := lib_driver_cmd_wilc1000

   Enable build lib_driver_cmd_wilc1000 in hardware/atmel/wilc1000/wpa_supplicant_8_lib.

   – WIFI_DRIVER_MODULE_NAME / WIFI_DRIVER_MODULE_PATH

   wifi.c install(insmod) or remove(rmmod) the defined Kernel module. The path could be different for platform.

   – WIFI_DRIVER_FW_PATH_XXX (**not used, definition only**)

   Firmware will be downloaded by the Kernel module automatically.

   – BOARD_WLAN_DEVICE := wilc1000

   This definition will be used for extra components or framework patch.

WARNING **SAMA5D3-EK** supports Realtek chipset for Wi-Fi/Bluetooth® module basically. Because ATWILC1000 may conflict with Realtek chipset, it is highly recommend to remove the part of RealTek in BoardConfig.mk. This is the same as on other Android platforms.

```
#BOARD_WIFI_VENDOR := realtek

ifeq ($(BOARD_WIFI_VENDOR), realtek)

    WPA_SUPPLICANT_VERSION := VER_0_8_X

    BOARD_WPA_SUPPLICANT_DRIVER := NL80211

    …

    WIFI_DRIVER_FW_PATH_P2P   := ""

    WIFI_DRIVER_FW_PATH_PARAM := ""

endif
```

2. Product package definitions.

    The ATWILC1000 driver requires firmware binaries, and the ATWILC1000 software package provides firmware in the "hardware/atmel/wilc1000/firmware". To add this packages in system image, declare the "PRODUCT_PACKAGE" in the product mk file.

    The location and name of product mk file is depends on the platform.

    device / <vendor-name> / <device-name> / xxx.mk

    The product "mk" file can be found by tracing from "AndroidProducts.mk" file.

INFO In **SAMA5D3-EK**, "device/atmel/common/config/Android_Copy.mk" is used. This file is inherited from "device/atmel/sama5d3/device.mk".

```
PRODUCT_PACKAGES += \

        wilc1000_fw.bin \

        wilc1000_ap_fw.bin \

        wilc1000_p2p_fw.bin
```

3. Launch Wi-Fi services.

    To launch wpa_supplicant daemon and dhcpcd, set the following in "init.xxx.rc".

    The location and name of "init.xxx.rc" file is depends on the platform.

    device / <vendor-name> / <device-name> / init.xxx.rc

TIPS Below settings can be written separately in multiple files.

INFO In **SAMA5D3-EK**, the "init.rc" and "init.sama5-pda.rc" are used. They are located in the "device/atmel/sama5d3/".

– **Permissions and paths**

```
mkdir /data/misc/wifi 0770 wifi wifi

chmod 0660 /data/misc/wifi/wpa_supplicant.conf

mkdir /data/misc/wifi/sockets 0770 wifi wifi

mkdir /data/misc/dhcp 0770 dhcp dhcp

chown dhcp dhcp /data/misc/dhcp
```

– **wpa_supplicant only for Station mode**

```
service wpa_supplicant /system/bin/wpa_supplicant \

    -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
```

TIPS   To run wpa_supplicant daemon, the wpa_supplicant.conf file should be located in "/data/misc/wifi/". This file is generated by hardware/atmel/wilc1000/config/Android.mk.

– **wpa_supplicant for concurrent mode**

```
service p2p_supplicant /system/bin/wpa_supplicant \

    -ip2p0 -Dnl80211 -c/data/misc/wifi/p2p_supplicant.conf -N \

    -iwlan0 -Dnl80211 -c/data/misc/wifi/wpa_supplicant.conf \

    -O/data/misc/wifi/sockets -puse_p2p_group_interface=1 \

    -e/data/misc/wifi/entropy.bin -g@android:wpa_wlan0

    class main

    socket wpa_wlan0 dgram 660 wifi wifi

    disabled

    oneshot
```

TIPS   To add Wi-Fi Direct feature, the "android.hardware.wifi.direct.xml" file is required. In this case, p2p_supplicant service will be invoked instead of wpa_supplicant service.

– **dhcpcd**

```
service dhcpcd_wlan0 /system/bin/dhcpcd -aABDKL

    class main

    disabled

    oneshot


service dhcpcd_p2p /system/bin/dhcpcd -aABKL

    class main

    disabled

    oneshot


service iprenew_wlan0 /system/bin/dhcpcd -n

    class main

    disabled

    oneshot


service iprenew_p2p /system/bin/dhcpcd -n

    class main

    disabled

    oneshot
```

4. Extra components for ATWILC1000.

    ATWILC1000 specific parts are located in hardware/atmel/wilc1000.

    GitHub Android ATWILC1000 Site provides this part: https://github.com/android4sc/kitkat.git.

    To apply this section, `BOARD_WLAN_DEVICE := wilc1000` should be defined in BoardConfig.mk.

```
└─wilc1000
        Android.mk

    ┌─config
    │       Android.mk
    │
    ┌─firmware
    │       Android.mk
    │       wilc1000_ap_fw.bin
    │       wilc1000_fw.bin
    │       wilc1000_p2p_fw.bin
    │
    └─wpa_supplicant_8_lib
            Android.mk
            driver_cmd_nl80211.c
            MODULE_LICENSE_BSD
            NOTICE
```

– **wpa_supplicant.conf**

hardware / atmel / wilc1000 / config /

wpa_supplicant.conf is used for executing wpa_supplicant service. Android.mk in this folder invokes the wpa_supplicant_conf.mk to generation wpa_supplicant.conf as the version of wpa_supplicant. In this time, wpa_supplicant.conf file is located in "/system/etc/wifi". At the actual operating time, wifi.c will copy wpa_supplicant.conf to "/data/misc/wifi/".

– **firmware**

hardware / atmel / wilc1000 / firmware /

There are three kinds of firmware:

- Station mode: wilc1000_fw.bin
- AP mode: wilc1000_ap_fw.bin
- P2P mode: wilc1000_p2p_fw.bin

ATWILC1000 Kernel module download these files as action mode.

– **Private library for wpa_supplicant**

hardware / atmel / wilc1000 / wpa_supplicant_8_lib /

Android uses a vendor specific library for implementing driver interface function. This section builds the "lib_driver_cmd_wilc.a" library including the wrapper code to allow this mechanism.

5. HAL patch for ATWILC1000.

– "wifi.c" patch

hardware / libhardware_legacy / wifi /

```
└─wifi
        Android.mk
        wifi.c
```

- To patch for ATWILC1000 with "WILC1000_WIFI_USED" definition in "wifi.c" file, add below flags into Android.mk

```
ifeq ($(BOARD_WLAN_DEVICE), wilc1000)

LOCAL_CFLAGS += -DWILC1000_WIFI_USED

endif
```

- Java frameworks expects the firmware-loader. But ATWILC1000 package need not firmware-loader. Instead, the Kernel module downloads the firmware by itself.

```
int wifi_change_fw_path(const char *fwpath)

{

    int len;

    int fd;

    int ret = 0;


#ifndef WILC1000_WIFI_USED

    if (!fwpath)

        return ret;

    fd = TEMP_FAILURE_RETRY(open(WIFI_DRIVER_FW_PATH_PARAM,
O_WRONLY));

    if (fd < 0) {

        ALOGE("Failed to open wlan fw path param (%s)",
strerror(errno));

        return -1;

    }

    len = strlen(fwpath) + 1;

    if (TEMP_FAILURE_RETRY(write(fd, fwpath, len)) != len) {

        ALOGE("Failed to write wlan fw path param (%s)",
strerror(errno));

        ret = -1;

    }

    close(fd);
#endif

    return ret;

}
```

- After inserting ATWILC1000 Kernel module with *insmod* function, it is necessary to wait until the wlan0 interface is ready.

```
#define TIME_COUNT 20 // 200ms*20 = 4 seconds for completion
int wifi_load_driver()
{
…
    if (strcmp(FIRMWARE_LOADER,"") == 0) {
#ifdef WILC1000_WIFI_USED
            char tmp_buf[200] = {0};
            FILE *profs_entry = NULL;
            int try_time = 0;
            do {
                    profs_entry = fopen("/proc/net/wireless",
"r");
                    if(profs_entry == NULL){
                            ALOGE("open /proc/net/wireless
failed!");
                            property_set(DRIVER_PROP_NAME,
"failed");
                            break;
            }

            if( 0 == fread(tmp_buf, 200, 1, profs_entry) ){
                    ALOGD("faied to read proc/net/wireless");
            }

            if(NULL != strstr(tmp_buf, "wlan0")) {
                    ALOGD("insmod okay,try_time(%d)", try_time);
                    fclose(profs_entry);
                    profs_entry = NULL;
                    property_set(DRIVER_PROP_NAME, "ok");
                    break;
            }else {
                    ALOGD("initial,try_time(%d)",try_time);
                    property_set(DRIVER_PROP_NAME, "failed");
            }
            fclose(profs_entry);
            profs_entry = NULL;
            usleep(200000);
            }while(try_time++ <= TIME_COUNT);// 4 seconds
#else
            /* usleep(WIFI_DRIVER_LOADER_DELAY); */
            property_set(DRIVER_PROP_NAME, "ok");
#endif
 }
…
```

8.3   Build Android

1.   Configure and build Android.

```
$ . build/envsetup.sh

$ lunch sama5d3-eng

$ make
```

2.   Generate Android Image for booting from NAND flash.

```
$ mkubi_image -b sama5d3
```

RESULT   If successfully, "system_ubifs-SAMA5D3-ANDROID-4.4.2_r2.img" and
"userdata_ubifs-SAMA5D3-ANDROID-4.4.2_r2.img" will be generated in <an-
droid_working_dir>/.

9   Linux Kernel Programming under Android

9.1   Get Source Code

Get Android Linux Kernel source by the following command.

```
$ git clone git://github.com/Android4SAM/linux-at91.git

$ cd linux-at91

$ git checkout -b linux-at91 Android4sam_v4.4
```

TIPS   The manifest repository URL and branch could be updated. It is highly recommended to visit the Atmel Android for SAM site to check the latest status.

WARNING   Android may not be downloaded due to your internet settings. Check the connection and firewall status.

9.2   Patching ATWILC1000 Driver Source

WARNING   Reference patches are based on the SAMA5D3-EK, Chapter 2 depicts the step-by-step core integration.

1. Kernel Configuration.

   In **SAMA5D3-EK**, "sama5_android_defconfig" is used.

2. Customizing ATWILC1000 source for SPI.

   To register SPI slave device, Device ID must be added in the DTS (Device Tree Source) file. For the reference of **SAMA5D3-EK**, the below files are to be modified by using SPI1.

   – sama5d36ek_pda7.dts

```
ahb {
            apb {
                    spi0: spi@f0004000 {
                            status = "okay";
                    };

                    spi1: spi@f8008000 {
                            status = "okay";
                    };
```

   – sama5d3xcm.dtsi

```
ahb {
        apb {
                spi0: spi@f0004000 {
                        cs-gpios = <&pioD 13 0>, <0>, <0>, <0>;
                };

                spi1: spi@f8008000 {
                        cs-gpios = <&pioC 25 0>, <0>, <0>, <&pioD 16 0>;
                };
        };
```

– sama5d3xmb.dtsi

```
ahb {
        apb {
                …
                spi0: spi@f0004000 {
                        m25p80@0 {
                                compatible = "atmel,at25df321a";
                                spi-max-frequency = <50000000>;
                                reg = <0>;
                        };
                };

                spi1: spi@f8008000 {
                        wilc_spi@0 {
                                compatible = "atmel,wilc_spi";
                                spi-max-frequency = <6000000>;
                                reg = <0>;
                        };
                };
```

3. Customizing ATWILC1000 source for interrupt.

In **SAMA5D3-EK**, the PC4 pin is already used for Ethernet port. It is highly recommended to use the other one, such as PC28 (gpio number 92). In the case of using PC28 for external interrupt, modify the driver source, "drivers/net/wireless/atmel/wilc/linux_wlan_common.h" as below:

```
#define GPIO_NUM (92)    /* for the PIOC28 */
```

TIPS   To apply modified device-tree to the device, compile and download the dtb (Device Tree Blob) file to the target.

Atmel

9.3   Build Linux Kernel

⚠️ **WARNING**   Before you build Android Linux Kernel, you need to copy the root (<android_work-ing_dir> / out / target / product / sama5d3 / root) folder build from Android to the An-droid Linux Kernel root folder.

```
$ make mrproper

$ make ARCH=arm sama5_android_defconfig

$ make ARCH=arm CROSS_COMPILE=(path_to_cross-compiler/cross-compiler-prefix-) zImage

$ make ARCH=arm CROSS_COMPILE=(path_to_cross-compiler/cross-compiler-prefix-) modules

$ make ARCH=arm CROSS_COMPILE=(path_to_cross-compiler/cross-compiler-prefix-) dtbs
```

9.4   Copy Kernel Module into File System (only for SAMAD3-EK)

ℹ️ **INFO**   Usually, the Android build system automatically include Kernel modules into system image. But in **SAMA5D3-EK**, the Kernel module should be copied manually.

If you need to update ATWILC1000 Kernel module:

1. Copy new ATWILC1000 module.
   – From <linux_working_dir>/drivers/net/wireless/atmel/wilc1000/wilc1000.ko
   – To the Android device (<android_working_dir>/device/atmel/common/config/) folder.

2. Add the below lines into "device/atmel/common/config/Android_Copy.mk".

```
PRODUCT_COPY_FILES += \

        $(LOCAL_PATH)/wilc1000.ko:system/lib/modules/wilc1000.ko
```

3. Build Android and generate Android image again.

10 How to Update Binary on SAMA5D3-EK

10.1 Install SAM-BA Tool

SAM-BA is a software that provides a way to easily program different Atmel AT91SAM devices. It is integrated in the AT91 In-system Programmer (ISP for short). This latest ISP application can be found on the Atmel official website:

http://www.atmel.com/tools/ATMELSAM-BAIN-SYSTEMPROGRAMMER.aspx

You can choose the proper version for your OS and also USB CDC driver.

10.2 Preparing Binary

To integrate the ATWILC1000 package, only updating the system image and Kernel image is required. In some cases, such as the SPI mode, the device-tree may also be updated.

- zImage
- system_ubifs-SAMA5D3-ANDROID-4.4.2_r2.img
- sama5d36ek_pda7.dtb

The others, such as the bootstrap and u-boot image, you can use prebuilt image provided in Demo Package section of Atmel Android for SAM website with the exception of the above files to make the story simpler:

http://www.at91.com/android4sam/bin/view/Android4SAM/Sama5d3xek

This Demo Package includes the batch file and shell script to run the SAM-BA tool.

10.3 Downloading Image

1. For full Demo package go to Atmel Android for SAM Site: http://www.at91.com/android4sam.
2. Replace the marked files in the marked path below.
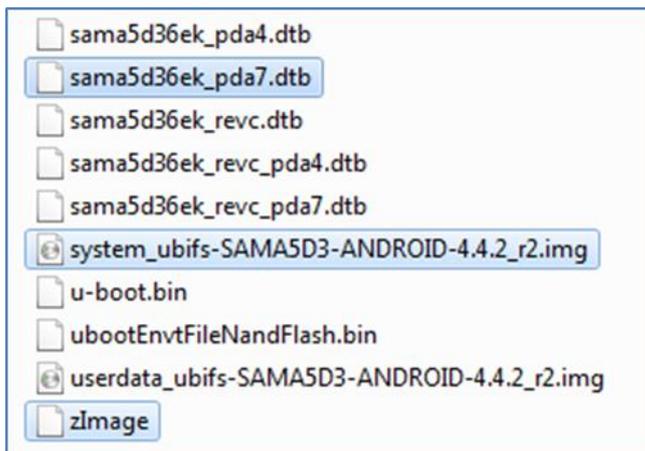
- – Kernel Image: from <linux_working_dir>/arm/boot/zImage
- – System Image   from <android_working_dir> /system*.img
- – Device-tree Blob: from <linux_working_dir>/arm/boot/dts/sama5d36ek_pda7.dtb

3. When connected the first time the USB driver must be installed. This driver is provided with the SAM-BA tool.

Atmel
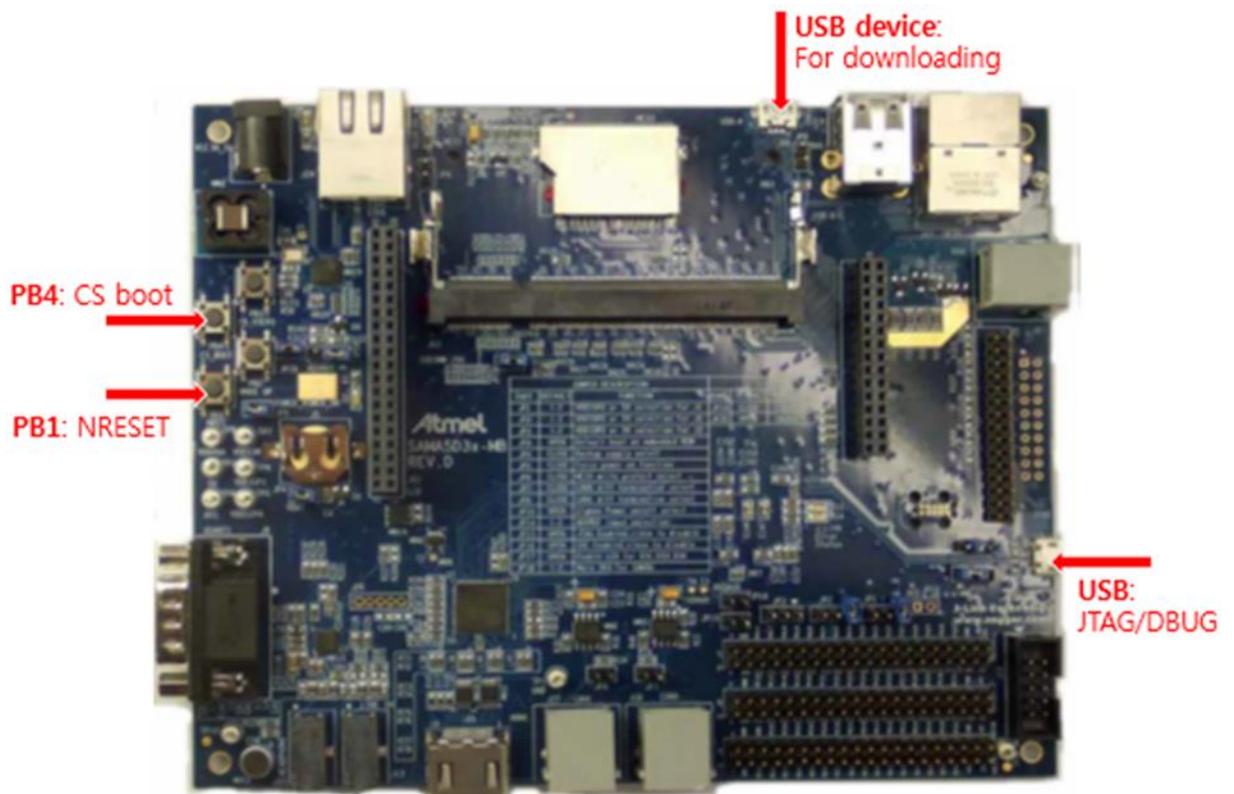
4. In order to boot from SAM-BA, keep pressing the pushbutton PB4 (CS boot disable) and perform the reset. A reset can be forced by pressing the pushbutton PB1 (NRST).

5. There are many batch files (Windows) and shell scripts (Linux) in the Demo Package. Choose the proper file as your host OS and target platform. In case of SAMA5D3-EK with TM7000 display module and Windows system, run the "sama5d3ek_pda7_nandflash.bat".

6. You can see the log message via USB port marked as "JTAG".

11   Revision History

Doc Rev.	Date	Comments
42435A	03/2015	Initial document release.