LAIRD ANDROID SD45 SOFTWARE INTEGRATION GUIDE

Application Note *v1.8*

OVERVIEW

This document explains the steps required to fully integrate the SD45 Android software package.

Laird recommends that you thoroughly analyse each step of the process. Each individual step should be integrated separately and manually tested. Attempting all of the steps at once without testing will likely cause bugs that are difficult to troubleshoot as a whole. By contrast, a step-by-step approach will ensure that each procedure is successful for the dependent steps that follow.

Integrating a Wi-Fi driver into specific Android platform code can be challenging and may require platform-specific changes. If the following information is confusing or does not provide the proper results, please visit https://laird-ews-support.desk.com/ for further assistance.

This guide covers integration of both Wi-Fi and Bluetooth functionality of the SD45 on Android.

Note: The SD45 is available as part of a bundle pack which contains both a 45 series radio and the Laird BT830. Specific information for integrating the BT830 is therefore provided in this guide in the section **BT830 Specific Integration**.

REQUIRED FILES

There are several files required for successful software integration. They are of two kinds: Open source drivers provided by Atheros, and binaries provided by Laird.

Atheros Open Source Driver Files

- ath6kl_core.ko
- ath6kl_sdio.ko

These two drivers must be built-in or cross-compiled against your Linux kernel. See Compiling the ath6kl Driver and Configuring a Linux-Backports Release for more information.

Note: From Linux 3.2 onward, the source code necessary to compile Atheros drivers is natively included as a proper wireless driver. See http://wireless.kernel.org/en/users/Drivers/ath6kl for more information and access to linux-backports releases of the driver.

Laird recommends the use of linux-backports to cross-compile the latest open source version of the ath6kl driver. This ensures access to the newest upstream source changes, regardless of your platform's kernel version. See the Configuring a Linux-backports Release for further instructions.

Laird-Provided Binary Files

If this is a new integration, and you have not already been provided the following files, please contact ews.support@lairdtech.com or wireless.support@lairdtech.com for the following files.

- **Atheros** firmware **binary**

  fw-3.bin (or fw-4.bin with newer Atheros Wi-Fi drivers)

  The open source driver can load the publically available firmware from Atheros. However, this should not be used for anything other than debugging purposes. The firmware file provided by Laird includes support for enhanced roaming algorithms, as well as other proprietary functionality that does not work with the public firmware.

  Note: Laird does not support any version of firmware other than the one specifically provided by Laird.

- **Atheros board calibration file**

  bdata.bin

  The board calibration file is tuned for use with our SD45 module. To maintain regulatory compliance, the bdata.bin file provided by Laird **MUST** be used.

  Note: Laird does not support any other board calibration files.

- **Laird Android wpa_supplicant**

  wpa_supplicant.bin

  Laird provides their own proprietary wpa_supplicant which supports enhanced encryption methods, along with CCX support. This supplicant is also used in conjunction with the Android LCM to configure the Wi-Fi connection.

  Note: The MSD45 may be alternatively used with the wpa_supplicant_8 included inside of the AOSP; however, Laird does not directly support this.

- **Laird's Connection Manager Application**

  LCM.apk

  Laird provides their own proprietary Android application to control Laird-specific parameters. See Installing the LCM.

  Note: This application may be used in tandem with the native Android connection manager.

COMPILING THE ATH6KL DRIVER

The following flags must be set in the Linux kernel's .config file:

- **Enable Wireless and CFG80211**

  CONFIG_WIRELESS=y
  CONFIG_CFG80211=m
  CONFIG_WLAN=y

- **Atheros Specific**

  CONFIG_ATH_COMMON=m
  CONFIG_ATH6KL=m
  CONFIG_ATH6KL_SDIO=m

- **Optional/Debugging**

  CONFIG_ATH_DEBUG=y
  CONFIG_ATH6KL_DEBUG=y

Note: Additional .config flags are device specific and may be required for proper compilation. These flags are merely the minimum of what is required to compile the ath6kl drivers.

In newer Linux kernel's, CONFIG_ATH_COMMON has been replaced with CONFIG_ATH_CARDS. The functionality and usage between the two is identical.

Flags set in this manner create ath6kl_core and ath6kl_sdio.ko files. Optionally, the driver can also be compiled with the 'y' flag to incorporate the drivers directly into the kernel. This can be done for ath6kl_core or ath6kl_sdio.

Example: CONFIG_ATH6KL_SDIO=y

Note: It is impossible to load externally cross-compiled linux-backports releases if they are included natively in the kernel.

On some Android platforms, the SDIO bus speed is, by default, set to 50 MHz or greater. For initial integration, we suggest that you configure this to 25 MHz. This must be specifically defined in the Linux kernel.

CONFIGURING A LINUX-BACKPORTS RELEASE

Required Files

The latest stable Linux-backports release is located at:
https://www.kernel.org/pub/linux/kernel/projects/backports/stable/

Be sure to download and un-zip the file before proceeding.

Build Process

Linux backports usage documentation can be found at:
https://backports.wiki.kernel.org/index.php/Documentation#Usage_guide

Be sure to go over the section regarding Cross compiling for proper creation of the ath6kl driver.

Note: Ignore the *make install* command; it does not work properly for this situation. The final step is *make* (rather than *make install*).

```
Feel free to use "make oldconfig" to manually configure the .config
appropriately. When complete, the following .config values should be set:
```

```
CPTCFG_WIRELESS=y
CPTCFG_NET_CORE=y
CPTCFG_EXPERT=y
...
CPTCFG_CFG80211=m
CPTCFG_CFG80211_DEFAULT_PS=y
CPTCFG_CFG80211_WEXT=y
...
```

```
CPTCFG_WLAN=y
...
CPTCFG_ATH_CARDS=m
CPTCFG_ATH_DEBUG=y
CPTCFG_ATH6KL=m
CPTCFG_ATH6KL_SDIO=m
CPTCFG_ATH6KL_DEBUG=y
```

Generated Files

The following generated files must be copied into the Android File System:

- ath6kl_core.ko – Can be found at /drivers/net/wireless/ath/ath6kl
- ath6kl_sdio.ko – Can be found at /drivers/net/wireless/ath/ath6kl
- compat.ko – Can be found at /compat/compat.ko
- cfg80211.ko – Can be found at /net/wireless/cfg80211.ko

Adding Files to the Android File System

Required Files

Create a folder that contains the modules to be copied, along with an .mk file with the following entries:

```
ifeq ($(BOARD_WLAN_DEVICE), qcwcn)
PRODUCT_COPY_FILES += \
  $(LOCAL_PATH)wpa_supplicant:system/bin/wpa_supplicant \
  $(LOCAL_PATH)wpa_cli:system/bin/wpa_cli \
  $(LOCAL_PATH)wpa_supplicant.conf:system/etc/wifi/wpa_supplicant.conf \
  $(LOCAL_PATH)/fw_v3.4.0.81.bin:system/vendor/firmware/ath6k/AR6003/hw2.1.1/fw-3.bin \
  $(LOCAL_PATH)/bdata.bin:system/vendor/firmware/ath6k/AR6003/hw2.1.1/bdata.bin \
  $(LOCAL_PATH)cfg80211.ko:system/lib/modules/cfg80211.ko \
  $(LOCAL_PATH)compat.ko:system/lib/modules/compat.ko \
  $(LOCAL_PATH)ath6kl_core.ko:system/lib/modules/ath6kl_core.ko \
  $(LOCAL_PATH)ath6kl_sdio.ko:system/lib/modules/ath6kl_sdio.ko
endif
```

Android has its own default location to look for vendor firmware. The path is defined in /system/core/init/devices.c

Note: AOSP Jellybean source defines three locations which are version specific:

```
#define FIRMWARE_DIR1    "/etc/firmware"
#define FIRMWARE_DIR2    "/vendor/firmware"
#define FIRMWARE_DIR3    "/firmware/image"
```

The open source Atheros driver always looks for the firmware and board calibration files in the following location: **ath6k/AR6003/hw2.1.1**

For Laird's integration, FIRMWARE_DIR2 was selected. The files are copied into the following location: **/vendor/firmware/ath6k/AR6003/hw2.1.1**.

Laird selected **system/lib/modules** as a logical location for the seperate kernel object files. However, any Android file system location with the correct permissions may be used. Further documentation below describes how to configure /system/lib/modules to have the appropriate permissions.

INSTALLING LCM.APK

Adding as a System Application

To give the LCM application appropriate privileges so that it can correctly exercise Wi-Fi functionality, it must be added as an Android System Application.

All that is required for this installation is to make sure the LCM.apk file is copied to: **/system/app/LCM.apk**

MODIFYING BOARDCONFIG.MK

Note: Many chip vendors provide a BoardConfig.mk file that already includes a reference to another Wi-Fi module. Duplicate definitions cause problems with Wi-Fi integration. Prior to starting, please comment out (or completely remove) unnecessary Wi-Fi card definitions.

Required

Add the following to the end of your BoardConfig.mk file:

```
BOARD_WIFI_VENDOR := atheros


ifeq ($(BOARD_WIFI_VENDOR), atheros)
    BOARD_WLAN_DEVICE       := qcwcn
    BOARD_HAS_ATH_WLAN      := true
endif
```

Note: Laird provides a pre-compiled wpa_supplicant.bin, so wpa_supplicant and wpa_supplicant_drv definitions are not required.

Installing .ko Files

If you have chosen to compile the Atheros driver (drivers) as .ko files, you must define the following in the .mk file.

```
WIFI_DRIVER_MODULE_PATH := "/system/lib/modules/ath6kl_sdio.ko"
WIFI_DRIVER_MODULE_NAME := "ath6kl_sdio"
```

Note: Android requires that DRIVER_MODULE_PATH and WIFI_DRIVER_MODULE_NAME are defined at least once inside of BoardConfig.mk. If they are not defined, the insmod section of wifi.c will not be configured correctly.

If both ath6kl_core and ath6kl_sdio are compiled as .ko modules, only the ath6kl_sdio.ko should be referenced in the BoardConfig.mk. In this scenario, the insmod for ath6kl_core.ko must be added to the Wi-Fi HAL code. Please see Modifying Wifi.c for more information.

MODIFYING WIFI.C

Note: The following changes may be optional depending on your configuration. They may also be optional depending on the version of Android that is being used. Please read notes below for more information.

The information below relates to modifying the HAL (Hardware Abstraction Layer) file:
<platform>/hardware/libhardware_legacy/wifi/wifi.c

Integration Type 1

- Both ath6kl_core and ath6kl_sdio are built into the Linux kernel.

When the modules are built directly into the kernel, wifi.c no longer needs to insmod or rmmod a specific driver. Therefore, these two methods should be commented out and instead immediately return 0.

For example, the following modified code snippet was taken from Jellybean's version of wifi.c:

```
static int insmod(const char *filename, const char *args)
{
    return 0;
// Module built into kernel.
/*
    void *module;
    unsigned int size;
    int ret;

    module = load file(filename, &size);
    if (!module)
        return -1;

    ret = init module(module, size, args);

    free(module);

    return ret;
*/
}
```

Integration Type 2

- ath6kl_core is built into the Linux kernel and ath6kl_sdio is built as a .ko file.

No insmod or rmmod changes to wifi.c are required.

Integration Type 3

- Both ath6kl_core and ath6kl_sdio are built as .ko files.

With no modifications, wifi.c only insmods ath6kl_sdio. This fails because ath6kl_sdio requires that ath6kl_core.ko is loaded. Therefore ath6kl_core.ko must be insmoded prior to (and rmmoded after) ath6kl_sdio.ko.

Furthermore, any additional files created with a linux-backport release – specifically compat.ko and cfg80211.ko must also be insmoded in the approprate order. See the following souce code example:

```
...
static const char COMPAT MODULE NAME[]          = "compat";
static const char CFG80211 MODULE NAME[]        = "cfg80211";
static const char CORE MODULE NAME[]            = "ath6kl core";
static const char COMPAT MODULE PATH[]          =
"/system/lib/modules/compat.ko";
static const char CFG80211 MODULE PATH[]        =
"/system/lib/modules/cfg80211.ko";
static const char CORE MODULE PATH[]            =
"/system/lib/modules/ath6kl core.ko";
...
static int insmod(const char *filename sdio, const char *args)
{
    void *module compat;
    void *module cfg;
    void *module core;
    void *module sdio;
    unsigned int size compat;
    unsigned int size cfg;
    unsigned int size core;
    unsigned int size sdio;
    int ret;

    // Loading modules
    module compat = load file(COMPAT MODULE PATH, &size compat);
    if (!module compat)
        return -1;

    module cfg = load file(CFG80211 MODULE PATH, &size cfg);
    if (!module cfg)
        return -1;

    module core = load file(CORE MODULE PATH, &size core);
    if (!module core)
        return -1;

    module sdio = load file(filename sdio, &size sdio);
    if (!module sdio)
        return -1;

    // Inserting modules
    ret = init module(module compat, size compat, args);
    if (ret < 0)
        return ret;

    ret = init module(module cfg, size cfg, args);
    if (ret < 0)
```

```
        return ret;

    ret = init_module(module_core, size_core, args);
    if (ret < 0)
        return ret;

ALOGD("Inserting module at \"%s\"", filename_sdio);

    ret = init_module(module_sdio, size_sdio, args);
    if (ret < 0)
        return ret;

    free(module_compat);
    free(module_cfg);
    free(module_core);
    free(module_sdio);

    return ret;
}

static int rmmod(const char *modname_sdio)
{
    int ret = -1;
    int maxtry = 10;

    // SDIO
    while (maxtry-- > 0) {
        ret = delete_module(modname_sdio, O_NONBLOCK | O_EXCL);
        if (ret < 0 && errno == EAGAIN)
            usleep(500000);
        else
            break;
    }

    if (ret != 0){
        ALOGD("Unable to unload driver module \"%s\": %s\n",
             modname_sdio, strerror(errno));
        return ret;
    }

    // Core
    ret = -1;
    maxtry = 10;

    while (maxtry-- > 0) {
        ret = delete_module(CORE_MODULE_NAME, O_NONBLOCK | O_EXCL);
        if (ret < 0 && errno == EAGAIN)
            usleep(500000);
        else
            break;
    }

    if (ret != 0){
        ALOGD("Unable to unload driver module \"%s\": %s\n",
             modname_core, strerror(errno));
        return ret;
    }
```

```
    // CFG80211
    ret = -1;
    maxtry = 10;

    while (maxtry-- > 0) {
        ret = delete module(CFG80211 MODULE NAME, O NONBLOCK | O EXCL);
        if (ret < 0 && errno == EAGAIN)
            usleep(500000);
        else
            break;
    }

    if (ret != 0){
        ALOGD("Unable to unload driver module \"%s\": %s\n",
             CFG80211 MODULE NAME, strerror(errno));
        return ret;
    }

    // Compat
    ret = -1;
    maxtry = 10;

    while (maxtry-- > 0) {
        ret = delete module(COMPAT MODULE NAME, O NONBLOCK | O EXCL);
        if (ret < 0 && errno == EAGAIN)
            usleep(500000);
        else
            break;
    }

    if (ret != 0){
        ALOGD("Unable to unload driver module \"%s\": %s\n",
             COMPAT MODULE NAME, strerror(errno));
        return ret;
    }

    return ret;
}
```

MODIFYING INIT.RC

init.rc is a file that contains Android specific Init Language which provides both generic and machine level initialization instructions. The following Wi-Fi specific changes are used to enable the wpa_supplicant and allow Wi-Fi the correct sockets of communication. If done correctly, occurs automatically once Wi-Fi is turned on in Android.

Notes: As was mentioned in BoardConfig.mk, many chip vendors provide an init.rc file that already includes references to other Wi-Fi modules or wpa_supplicants. Duplicate definitions cause problems with Wi-Fi integration. Prior to starting, please comment out (or completely remove) unnecessary Wi-Fi related definitions.

init.rc is commonly divided into the standard init.rc for non-machine level initialization and init.<machine_name>.rc for machine level init. For certain platform integrations, it may be a better solution to place these changes into init.<machine_name>.rc.

The following is the Wi-Fi specific section of Laird's init.rc file for a Jellybean specific platform which uses wpa_supplicant_8.

Note: The following is divided into sections for clarification purposes.

File System Permissions

```
on post-fs-data

# give system access to wpa supplicant.conf for backup and restore
   mkdir /data/misc/wifi 0770 wifi system
   mkdir /data/misc/wifi/sockets 0770 wifi system
   chmod 0770 /data/misc/wifi
   chmod 0770 /data/misc/wifi/sockets
   chmod 0660 /data/misc/wifi/wpa supplicant.conf
   chmod 0775 /data/misc/wifi/ipconfig.txt
   mkdir /data/local 0751 root root

   # For Use with Laird Android SDK
   mkdir /data/Laird 0777 system system

   mkdir /data/misc/dhcp 0770 dhcp dhcp
   chown dhcp dhcp /data/misc/dhcp
```

Note: It is very important that the above is followed as closely as possible. Specifically make sure that the line: **mkdir/data/Laird 0777 system** is added. This directory is leveraged for storage by both LCM and the Laird supplicant.

Services and Properties on Boot

```
on boot

    # Define the Wi-Fi and other props
    setprop wifi.interface "wlan0"
```

```
    setprop wlan.interface "wlan0"
    setprop wlan.driver.status "ok"

service wpa_supplicant /system/bin/wpa_supplicant -Dnl80211 -iwlan0 -
c/data/misc/wifi/wpa_supplicant.conf -e/data/misc/wifi/entropy.bin
    class main
    socket wpa_wlan0 dgram 660 wifi wifi
    disabled
    oneshot

service dhcpcd_wlan0 /system/bin/dhcpcd -ABKL
    class main
    disabled
    oneshot

service iprenew_wlan0 /system/bin/dhcpcd -n
    class main
    disabled
    oneshot
```

JELLYBEAN SPECIFIC – DISABLE P2P_SUPPLICANT:

In Jellybean, significant changes were made to wifi.c code and by default, it attempts to load both a wpa_supplicant service and a p2p_supplicant service. P2p is not currently supported with the open source version of this driver and should be disabled in Jellybean during integration.

The following code changes are needed to disable p2p in Jellybean:

Modify the wpa_supplicant's android.config

In external/wpa_supplicant_8/wpa_supplicant/android.config, ensure that CONFIG_P2P, CONFIG_WIFI_DISPLAY, and CONFIG_AP are all = n (or are commented out).

See the following:

```
# Enable P2P

CONFIG_P2P=n

CONFIG_AP=n

#Enable Wifi Display
CONFIG_WIFI_DISPLAY=n
```

Modify device.mk

In your device.mk file (* this name can be platform specific), confirm that the *android.hardware.wifi.direct.xml* feature is **NOT** being copied into **out/.../system/etc/permissions**.

Specifically look for the following:

```
PRODUCT_COPY_FILES += \
```

```
...

frameworks/native/data/etc/android.hardware.wifi.direct.xml:system/etc/permi
ssions/android.hardware.wifi.
direct.xml
```

Remove config.xml References

In frameworks/base/core/res/res/values/config.xml remove the following references:

```
<item>"wifi_p2p,13,1,0,-1,true"</item>
      <bool translatable="false" name="config_wifi_p2p_support">true</bool>
```

BLUETOOTH – BASIC INTEGRATION

The following is required to integrate a Bluetooth radio into Android using the BlueZ stack.

Note: As of Android 4.2.2, Android has changed its native Bluetooth stack. From this point forward, BlueDroid source code is included by default in the AOSP.

Laird integration of the BlueDroid stack is still under development.

Bluetooth integration requires that you specify Bluetooth is on in BoardConfig.mk.

BoardConfig.mk

Confirm that the following is defined:

```
BOARD_HAVE_BLUETOOTH := true
```

BT830 SPECIFIC INTEGRATION

Laird's BT830 is available as a bundle purchase with SD45 radios as a way to prototype the interoperability between the two. For that reason, it may be required for some customers to additionally enable the BT830 in conjunction with the SD45.

The BT830 requires specific CSR PSKeys to be set prior to initialization. This can be configured through the BlueZ tool – bccmd. The following BlueZ commands will properly initialize the BT830 radio.

```
bccmd -t BCSP -d /dev/ttyS0 psload --reset defaults.psr
hciattach /dev/ttyS0 H4 115200
hciconfig hci0 up
```

The BlueZ stack can be started normally after this point.

REVISION HISTORY

Version	Date	Changes	Approved By
1.4	04 June 2014	Added links for clarity. Added an *Installing LCM.apk* section.	Brian Wagner
1.5	17 July 2014	Added "Bluetooth – Basic Integration" and "BT830-Specific Integration"	Brian Wagner
1.6	08 Aug 2014	Added note on SDIO Bus Clock speed to Modifying wifi.c	Brian Wagner
1.7	10 Dec 2014	Modified location of SDIO bus clock speed note. Added appropriate wpa_supplicant and wpa_supplicant.conf information for first-time integration.	Brian Wagner
1.8	15 Oct 2015	Added **Approved By** column to Rev History table	Sue White