Laird Android SD45/SD50 Software Integration Guide

Android, Jellybean, and KitKat

Application Note *v2.3*

Overview

This document explains the steps required to fully integrate the SD45 or SD50 Android software package into a device running Android.

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 or SD50 onto Android Jellybean.

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 BlueZ Bluetooth Integration section**.**

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
- ath6kl_usb.ko*

Note: The SD50 module supports both USB and SDIO interfaces. If the USB interface is being used, the correct driver interface will be ath6kl_usb.

These drivers must be built-in or cross-compiled against your Linux kernel. See 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**
  - SD45: fw-4.bin
  - SD50: fw.ram.bin

  The firmware file provided by Laird includes support for enhanced roaming algorithms, as well as other proprietary functionality that will not function with any other version of firmware not provided by Laird.

  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 radio module. To maintain regulatory compliance, the bdata.bin file provided by Laird **MUST** be used.

- **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 Android default wpa_supplicant_8 will function with the SD45 and SD50; however, Laird cannot support this supplicant. In addition, choosing to use this supplicant in place of Laird's supplicant will prevent any proprietary functionality from working properly.

- **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
  CONFIG_ATH6KL_USB=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.

COMPILING THE NATIVE WPA_SUPPLICANT

Even though Laird provides a binary wpa_supplicant for use, compiling the native platform wpa_supplicant is still required. This creates the appropriate libwpa for use with the platform.

Required Files

The wpa_supplicant build process requires a wpa_supplicant library. Download it at the following AOSP link:

https://android.googlesource.com/platform/hardware/qcom/wlan/

Add this code to **platform/external/**.

Required BoardConfiguration Values

Modify your platform's BoardConfig.mk with the following values:

```
BOARD_WLAN_DEVICE:      := qcwcn
BOARD_HAS_ATH_WLAN      := true
WPA_SUPPLICANT_VERSION := VER_0_8_X
BOARD_WPA_SUPPLICANT_DRIVER := NL80211
BOARD_WPA_SUPPLICANT_PRIVATE_LIB := lib_driver_cmd_qcwcn
```

CONFIGURING A LINUX-BACKPORTS RELEASE

Required Files

The latest stable Laird drop of Linux-backports is located at:
https://github.com/LairdCP/laird-linux-backports

Note:	Laird's backport tree was generated using backports against Laird's 4.1.1 kernel. A legacy 3.13 release is also available for older kernels.

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_USB=m
CPTCFG_ATH6KL_DEBUG=y
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

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
- ath6kl_usb.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:

Common Files:

```
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)/compat/cfg80211.ko:system/lib/modules/cfg80211.ko \

  $(LOCAL_PATH)/compat/compat.ko:system/lib/modules/compat.ko \

  $(LOCAL_PATH)/compat/ath6kl_core.ko:system/lib/modules/ath6kl_core.ko \

  $(LOCAL_PATH)/compat/ath6kl_usb.ko:system/lib/modules/ath6kl_usb.ko \

  $(LOCAL_PATH)/compat/ath6kl_sdio.ko:system/lib/modules/ath6kl_sdio.ko
endif
```

Additional SD45-Specific Files:

```
ifeq ($(BOARD_WLAN_DEVICE), qcwcn)
PRODUCT_COPY_FILES += \
  $(LOCAL_PATH)/fw_v3.4.0.84.bin:system/vendor/firmware/ath6k/AR6003/hw2.1.1/fw-4.bin \
  $(LOCAL_PATH)/bdata.bin:system/vendor/firmware/ath6k/AR6003/hw2.1.1/bdata.bin
endif
```

Additional SD50-Specific Files:

```
ifeq ($(BOARD_WLAN_DEVICE), qcwcn)
PRODUCT_COPY_FILES += \

$(LOCAL_PATH)/fw.ram.bin:system/vendor/firmware/ath6k/AR6004/hw3.0/fw.ram.
bin \
```

```
$(LOCAL_PATH)/bdata.bin:system/vendor/firmware/ath6k/AR6004/hw3.0/bdata.bi
n
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 SD45 driver expects the firmware and board calibration files in: **ath6k/AR6003/hw2.1.1**

The SD50 driver expects the firmware and board calibration files in: **ath6k/AR6004/hw3.0**

For Laird's integration, FIRMWARE_DIR2 was selected. The files are copied into the following location for SD45: **/vendor/firmware/ath6k/AR6003/hw2.1.1**. For an SD50 integration, the following location is used instead: **/vendor/firmware/ath6k/AR6004/hw3.0**.

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.** This can be accomplished natively with an android build the following way:

- Copy the LCM to <platform>/packages/apps/LCM
- Create the following entry in device.mk:
  `PRODUCT_PACKAGES += LCM`
- Inside of the <platform>/packages/apps/LCM, create an Android.mk with the following contents:

```
LOCAL_PATH:= $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE_TAGS := optional
LOCAL_MODULE := LCM
LOCAL_CERTIFICATE := PRESIGNED
LOCAL_SRC_FILES := LCM.apk
LOCAL_MODULE_CLASS := APPS
LOCAL_MODULE_SUFFIX := $(COMMON_ANDROID_PACKAGE_SUFFIX)
include $(BUILD_PREBUILT)
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

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.

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.

WIFI.C CHANGES

As we are currently configured, Wifi.c will only insmod ath6kl_sdio by default. The ath6kl_sdio kernel module alone will fail to insert, as it requires that ath6kl_core was inserted first.

Furthermore, any additional files created with a linux-backport release (specifically compat.ko and cfg80211.ko must also be inserted in the appropriate order. For this, Laird provides a wifi.c file inside of the hardware/libhardware_legacy/wifi/ directory which should be used. Additionally Laird include the appropriate versions of wifi.h and wifi_hal.h that correspond to this version of wifi.c.

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.<platform>.rc for machine level init. Laird recommends that the wpa_supplicant piece is added to the init.<platform>.rc file.

The following are the Wi-Fi specific sections of Laird's init.rc file for Jellybean and Kitkat respectively.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

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

File System Permissions

It is very important that the above is followed as closely as possible. Specifically make sure that the following line is added:

```
mkdir/data/Laird 0777 system
```

This directory is leveraged for storage by both LCM and the Laird supplicant.

Init.rc file Changes and Additions - Jellybean

```
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

…

on boot

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

Init.rc file Changes and Additions - KitKat

```
on post-fs-data

# give system access to wpa_supplicant.conf for backup and restore
    mkdir /system/etc/wifi 0770 wifi wifi
    chmod 0770 /system/etc/wifi
    chmod 0660 /system/etc/wifi/wpa_supplicant.conf
    chown wifi wifi /system/etc/wifi/wpa_supplicant.conf

    #wpa_supplicant
    mkdir /data/misc/wifi 0770 wifi wifi
    mkdir /data/misc/wifi/sockets 0770 wifi wifi
```

```
    chmod 0770 /data/misc/wifi
    chmod 0660 /data/misc/wifi/wpa_supplicant.conf
    chown wifi wifi /data/misc/wifi
    chown wifi wifi /data/misc/wifi/wpa_supplicant.conf

   # 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
…

on boot

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

Init.<platform>.rc File Changes and Additions - Jellybean

```
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
```

Init.<platform>.rc File Changes and Additions - KitKat

```
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

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

```
        class main
        disabled
        oneshot

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

DISABLE P2P_SUPPLICANT

Laird does not yet support the P2P supplicant. As such, the following configurations should be disabled prior to building Android:

Modify device.mk

In your device.mk file, confirm that the *android.hardware.wifi.direct.xml* feature is **NOT** being copied into **out/.../system/etc/permissions**.

Note:	The location of this file copy is not gaurenteed to be in device.mk for all Android platforms. Ultimately the most important thing is to be sure that the wifi.direct property is not being copied to system/etc/permissions/ in the out directory.

Specifically look for the following:

```
PRODUCT_COPY_FILES += \
...
frameworks/native/data/etc/android.hardware.wifi.direct.xml:system/etc/per
missions/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 INTEGRATION

Laird currently supports the BlueZ stack with our CSR Bluetooth radios and our SD50 radio on Android. The following steps outline the procedure to integrate this stack into Android 4.4 and 5.0.

Note:	As of Android 4.2.2, Android has changed its native Bluetooth stack to BlueDroid. Laird does not yet formally support this stack on Android. As such, the following instructions are based around the Android integration of the BlueZ stack.

These are tailored for integrating BlueZ into Android 4.4 KitKat and 5.0 Lollipop. If you are attempting to integrate BlueZ into Android Jellybean (4.2.2), you can use the version of BlueZ pre-packaged inside of Android 4.1.2.

Download the Appropriate BlueZ files (Android 4.4 and 5.0)

1. Remove the files currently located in the **platform/external/Bluetooth** directory. Then add the following references to your **platform/.repo/manifests/default.xml** file.

```
<remote name="gcode" fetch="https://code.google.com/p/"/>
<remote name="korg" fetch="git://git.kernel.org/pub/scm/bluetooth/"/>
<project remote="korg" path="external/bluetooth/bluez" name="bluez"
group="pdk" revision="master"/>
<project remote="korg" path="external/bluetooth/sbc" name="sbc"
group="pdk" revision="master"/>
<project remote="gcode" path="external/bluetooth/glib" name="aosp-
bluez.glib" group="pdk" revision="master"/>
```

2. Comment out all bdroid references in the default.xml file. After a repo sync, the required BlueZ bluetooth files is downloaded.

Patching Bionic

Android 5.0 provides all necessary bionic definitions. The following is only required if you are integrating against Android 4.4. The update work has already been done, and can be downloaded from the following link: https://github.com/bluez-android/aosp_platform_bionic/

Kernel Definitions / Backports

The following Kernel options should be enabled.

Note: BlueZ 5 requires the kernel to have management interface 1.3 or above. If your kernel is older than 3.9, you must add the following Bluetooth configurations to the backports creation process.

```
CONFIG_BT
CONFIG_BT_RFCOMM
CONFIG_BT_RFCOMM_TTY
CONFIG_BT_BNEP
CONFIG_BT_BNEP_MC_FILTER
CONFIG_BT_BNEP_PROTO_FILTER
CONFIG_BRIDGE
CONFIG_UHID
CONFIG_CRYPTO_CMAC
CONFIG_CRYPTO_USER_API
CONFIG_CRYPTO_USER_API_HASH
CONFIG_CRYPTO_USER_API_SKCIPHER
CONFIG_BT_HCIBTUSB
```

Necessary Tools

CSR radios require a few optional BlueZ tools for proper initialization. These can be enabled by adding or modifying the following entries inside of **external/bluetooth/bluez/android/Android.mk**.

```
#
# hcitool
#

include $(CLEAR_VARS)

LOCAL_SRC_FILES := \
     bluez/tools/hcitool.c \
     bluez/src/oui.c \
     bluez/lib/bluetooth.c \
     bluez/lib/hci.c \

LOCAL_C_INCLUDES := \
     $(LOCAL_PATH)/bluez \

LOCAL_CFLAGS := $(BLUEZ_COMMON_CFLAGS)

LOCAL_STATIC_LIBRARIES := \
     bluetooth-headers \

LOCAL_MODULE_PATH := $(TARGET_OUT_OPTIONAL_EXECUTABLES)
LOCAL_MODULE_TAGS := debug
LOCAL_MODULE := hcitool

LOCAL_ADDITIONAL_DEPENDENCIES := $(LOCAL_PATH)/bluez/configure.ac

include $(BUILD_EXECUTABLE)
#
# hciconfig
#

include $(CLEAR_VARS)

LOCAL_SRC_FILES:= \
     bluez/tools/hciconfig.c \
     bluez/tools/csr.c \
     bluez/lib/bluetooth.c \
     bluez/lib/hci.c \

LOCAL_C_INCLUDES := \
     $(LOCAL_PATH)/bluez \

LOCAL_CFLAGS := $(BLUEZ_COMMON_CFLAGS)

LOCAL_STATIC_LIBRARIES := \
     bluetooth-headers \

LOCAL_MODULE_PATH := $(TARGET_OUT_OPTIONAL_EXECUTABLES)
LOCAL_MODULE_TAGS := debug
LOCAL_MODULE := hciconfig
```

```
LOCAL_ADDITIONAL_DEPENDENCIES := $(LOCAL_PATH)/bluez/configure.ac

include $(BUILD_EXECUTABLE)
#
# hciattach
#

include $(CLEAR_VARS)

LOCAL_SRC_FILES := \
     bluez/tools/hciattach.c \
     bluez/tools/hciattach_st.c \
     bluez/tools/hciattach_ti.c \
     bluez/tools/hciattach_tialt.c \
     bluez/tools/hciattach_ath3k.c \
     bluez/tools/hciattach_qualcomm.c \
     bluez/tools/hciattach_intel.c \
     bluez/tools/hciattach_bcm43xx.c \
     bluez/lib/bluetooth.c \
     bluez/lib/hci.c \

LOCAL_C_INCLUDES := \
     $(LOCAL_PATH)/bluez \

LOCAL_CFLAGS := $(BLUEZ_COMMON_CFLAGS)

LOCAL_STATIC_LIBRARIES := \
     bluetooth-headers \

LOCAL_MODULE_PATH := $(TARGET_OUT_OPTIONAL_EXECUTABLES)
LOCAL_MODULE_TAGS := debug
LOCAL_MODULE := hciattach

LOCAL_ADDITIONAL_DEPENDENCIES := $(LOCAL_PATH)/bluez/configure.ac

include $(BUILD_EXECUTABLE)
#
# bccmd
#
include $(CLEAR_VARS)

LOCAL_SRC_FILES:= \
     bluez/lib/hci.c \
     bluez/lib/bluetooth.c \
     bluez/tools/bccmd.c \
     bluez/tools/csr.c \
     bluez/tools/csr_3wire.c \
     bluez/tools/csr_bcsp.c \
     bluez/tools/csr_hci.c \
     bluez/tools/csr_h4.c \
     bluez/tools/csr_usb.c \
     bluez/tools/ubcsp.c
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
LOCAL_CFLAGS := $(BLUEZ_COMMON_CFLAGS)

LOCAL_C_INCLUDES:=\
     $(LOCAL_PATH)/bluez \
     $(LOCAL_PATH)/bluez/lib \
     $(LOCAL_PATH)/bluez/tools \
     $(LOCAL_PATH)/bluez/src \
     $(LOCAL_PATH)/bluez/src/shared \

LOCAL_MODULE_PATH := $(TARGET_OUT_OPTIONAL_EXECUTABLES)
LOCAL_MODULE_TAGS := debug
LOCAL_MODULE:=bccmd

include $(BUILD_EXECUTABLE)
```

BoardConfig.mk

Confirm that the following is defined:

```
BOARD_HAVE_BLUETOOTH := true
```

Note: BLUETOOTH_BDROID references are no longer required after moving to BlueZ.

Init.rc

Add the following entries to your init.rc definition

```
import init.bluetooth.rc
on post-fs-data
    mkdir /data/misc/bluetooth 0770 bluetooth bluetooth
on boot
    start bt_init
#For Bluetooth
service bt_init /system/bin/logwrapper /system/bin/sh
/system/vendor/firmware/BlueZ/init.bt.sh
    class main
    user root
    group bluetooth net_bt_admin system
    disabled
    oneshot

on property:init.svc.bluetoothd=running
    start hci0_up

on property:init.svc.bluetoothd=stopped
    start hci0_down

service hci0_up /system/bin/logwrapper /system/xbin/hciconfig hci0 up
    class main
    group bluetooth
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
    disabled
    oneshot

service hci0_down /system/bin/logwrapper /system/xbin/hciconfig hci0 down
    class main
    group bluetooth
    disabled
    oneshot
```

BlueZ init script

The previous init.rc change specifically called on a shell script called 'init.bt'. The following is the contents of that file. It must be added to the Android filesystem. Laird recommends **/system/vendor**.

```
#!/system/bin/sh
LOG_TAG="Laird BlueZ Startup"
BTUART_PORT=/dev/ttyS2
PSCONFIG=/vendor/firmware/BlueZ/DWM-W311.psr
logi ()
{
  /system/bin/log -t $LOG_TAG -p i ": $@"
}
loge ()
{
  /system/bin/log -t $LOG_TAG -p e ": $@"
}
failed ()
{
  loge "$1: exit code $2"
  exit $2
}
# PS Config with bccmd
logwrapper /system/xbin/bccmd -t bcsp -d $BTUART_PORT -b 115200 psload -r
$PSCONFIG
case $? in
  0) logi "bccmd init port....done";;
  *) failed "port: $BTUART_PORT - bccmd failed" $?;
     exit $?;;
esac
# attach HCI
logwrapper /system/xbin/hciattach -p $BTUART_PORT bcsp 115200
case $? in
  0) logi "hci attached to : $BTUART_PORT";;
  *) failed "port: $BTUART_PORT - hciattach failed" $?;
     exit $?;;
esac
exit 0
```

Note: BTUART_PORT and PSCONFIG from the following script must be updated to the appropriate port/ location of the Bluetooth pskey file in your Android platform.

REVISION HISTORY

Version	Date	Notes	Approver
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 .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
2.0	7 Jan 2016	Created from SD45 Android Software Integration Guide v1.7. Added SD50 information and customized for Jellybean and KitKat requirements.	Brian Wagner
2.1	01 March 2016	Added wpa_supplicant build instructions Added 50 series BlueZ Bluetooth integration information	Brian Wagner
2.2	15 March 2016	Corrected a broken link in the **Overview**. Updated the link and note in the **Required Files** section. Removed outdated information in the **BoardConfig.mk** section.	Brian Wagner
2.3	27 Mar 2016	Formatting edits	Sue White

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610