GB86321G User Guide for Android

Revision History

Date	Version	Description	Author
2010/07/05	0.1	Initial revision of Bluetooth function 1. Only have instructions of how to enable Bluetooth 2. Bluetooth profile part will be included in next release	Andy Chang
2010/07/09	0.2	Initial revision of WLAN function 1. Driver version: 4.218.195.0	Terence Hsieh
2010/07/25	0.3	1. Refine the user guide 2. Add WLAN power saving mode	Fred Chen
2010/08/01	0.4	Add Bluetooth software architecture	Fred Chen
2010/08/03	0.5	1. Add Bluetooth power saving mode 2. Add Bluetooth test mode	Andy Chang
2010/08/18	0.6	Add SoftAP guide	Terence Hsieh
2010/08/18	0.7	Update Bluetooth “Wake Up from Sleep Mode”	Andy Chang
2010/10/15	0.8	Add Bluetooth Write Mac Address	Andy Chang
2010/11/16	0.9	Add WLAN deepsleep mode	Terence Hsieh
2011/01/07	1.0	1. Add auto channel selection of SoftAP 2. Add OOB (Out-Of-Band) mode	Terence Hsieh
2011/01/07	1.1	Add WLAN and BT coexistence enhancement	Terence Hsieh
2011/01/08	1.2	Correct typo in OOB mode	Fred Chen

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Ampak GB86321G module is a combo module which comprises wireless local area network (WLAN) and Bluetooth functions. This user guide is intended to give GB86321G users a general guide of how to enable the WLAN and Bluetooth functions in Android operating system.

Besides WLAN and Bluetooth basic functions, we'll also talk about the power saving mode for both WLAN and Bluetooth.

WLAN SOFTWARE ARCHITECTURE OVERVIEW

GB86321G WLAN DONGLE BASIC CONCEPT

The GB86321G WLAN software package contains the dongle host driver for the host, a downloadable binary image for the GB86321G, and management utilities.

The wireless driver runs on the GB86321G dongle. The SDIO host controller passes IEEE 802.3 packets, and the necessary control packets, back and forth over the SDIO bus. A special Broadcom Device Class protocol is used to encapsulate control packets on a separate logical control channel and to add packet information to the data channel.

The advantage of using the dongle concept is that the wireless driver is executed externally from a host device, which means the host device does not have to use CPU or memory resources in order to execute the wireless driver's functionality. The use of the dongle provides the following benefits to the host:

- Power savings
- A reduction in driver size and complexity
- Processor offloading for activities such as checksum calculation and Address Resolution Protocol (ARP) execution

WLAN DONGLE OVERVIEW

The Dongle Host Driver (DHD) is the executable module that provides encapsulated communication between the host device and the GB86321G module over the SDIO bus.

The dongle software architecture is based on two major components:

- Dongle Host Driver: A host-based driver used to provide a communication channel with the dongle device firmware.
- User-space configuration utilities, WL and DHD: These executable binaries are called “wl” and “dhd” respectively.

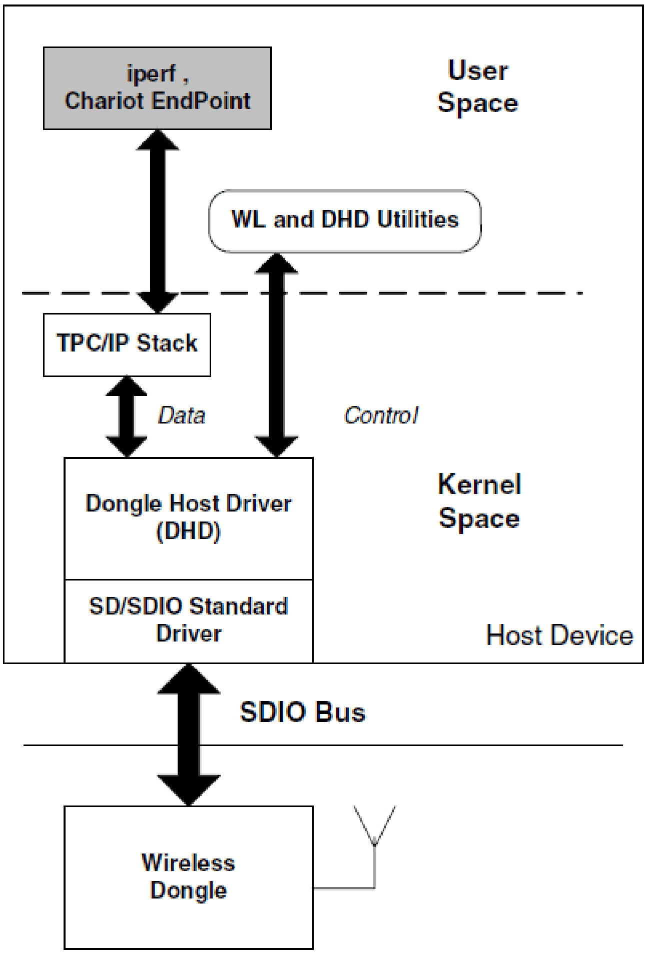


Figure 1: GB86321G SDIO WLAN Dongle Concept

WLAN SOFTWARE PACKAGE

The provided WLAN software package contains following files:

- Dongle host driver (dhd.ko)
- Dongle device firmware (sdio-g-cdc-full11n-reclaim-roml-wme.bin)
- User space configuration utility (dhd and wl)
- nvram.txt

WLAN DRIVER INSTALLATION

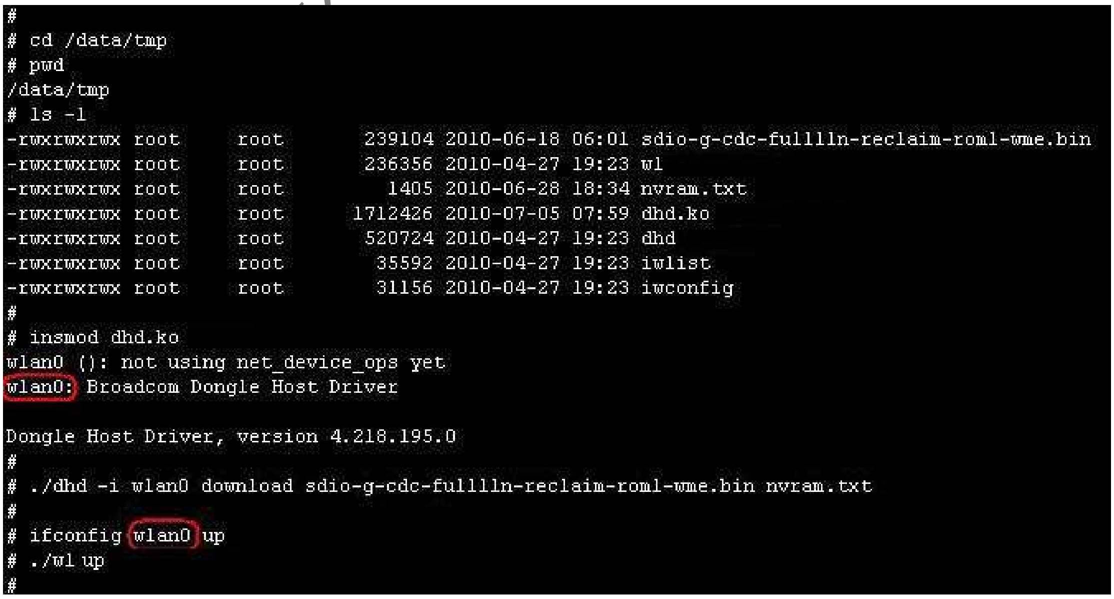
ENABLE WIRELESS EXTENSION OF LINUX KERNEL

Please add following items into your kernel configuration:

```
CONFIG_NET=y
CONFIG_WIRELESS=y
CONFIG_CFG80211=y
CONFIG_NL80211=y
CONFIG_WIRELESS_EXT=y
CONFIG_WIRELESS_EXT_SYSFS=y
```

INSTALL DONGLE HOST DRIVER

```
# insmod dhd.ko
# ./dhd -i [Interface Name] download sdio-g-cdc-full11n-reclaim-roml-wme.bin
   nvram.txt
# ifconfig [Interface Name] up
# ./wl up
```



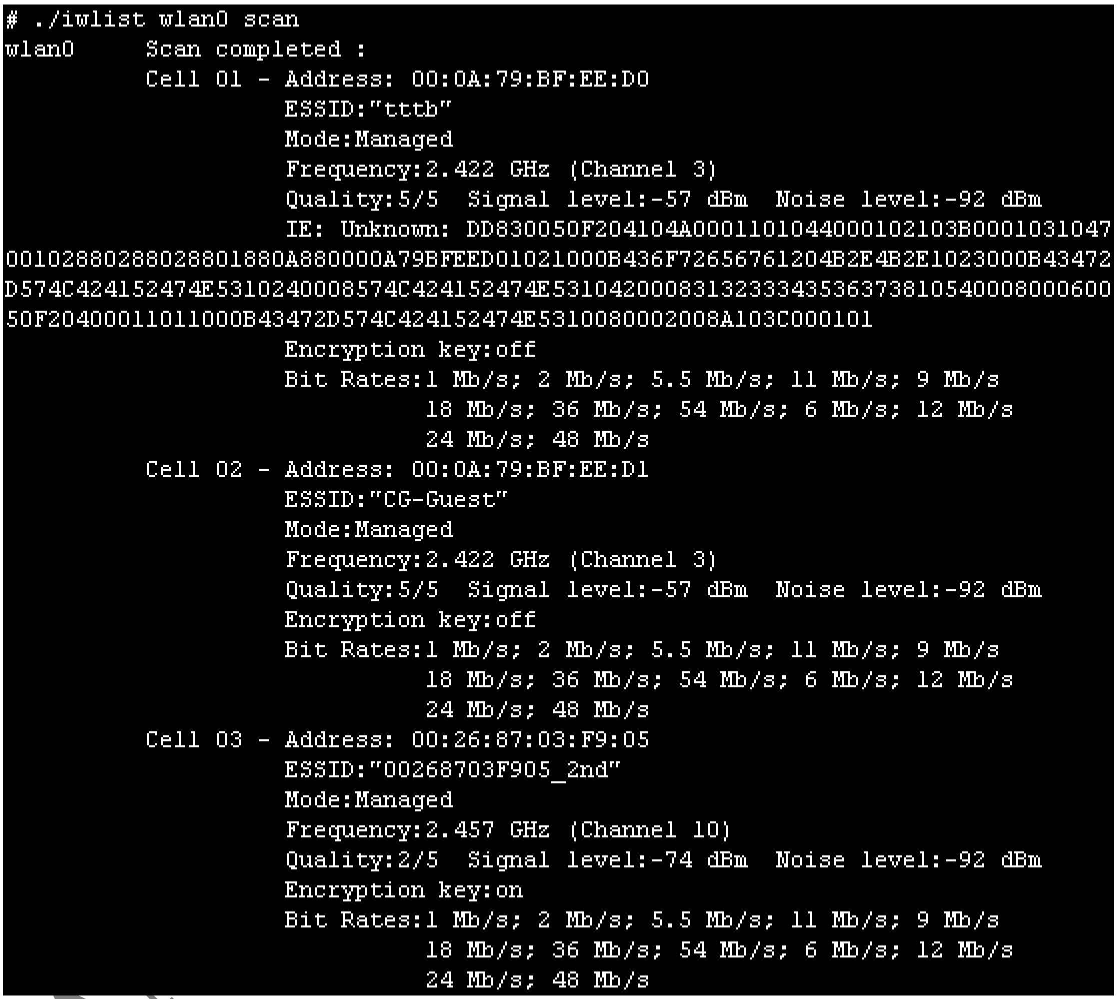
```
#
# cd /data/tmp
# pwd
/data/tmp
# ls -l
-rwxrwxrwx root     root       239104 2010-06-18 06:01 sdio-g-cdc-full11n-reclaim-roml-wme.bin
-rwxrwxrwx root     root       236356 2010-04-27 19:23 wl
-rwxrwxrwx root     root         1405 2010-06-28 18:34 nvram.txt
-rwxrwxrwx root     root      1712426 2010-07-05 07:59 dhd.ko
-rwxrwxrwx root     root       520724 2010-04-27 19:23 dhd
-rwxrwxrwx root     root        35592 2010-04-27 19:23 iwlist
-rwxrwxrwx root     root        31156 2010-04-27 19:23 iwconfig
#
# insmod dhd.ko
wlan0 (): not using net_device_ops yet
wlan0: Broadcom Dongle Host Driver

Dongle Host Driver, version 4.218.195.0
#
# ./dhd -i wlan0 download sdio-g-cdc-full11n-reclaim-roml-wme.bin nvram.txt
#
# ifconfig wlan0 up
# ./wl up
#
```

Figure 2: Install dongle host driver

WLAN OPERATION

SCAN NETWORK

#./iwlist [Interface Name] scan

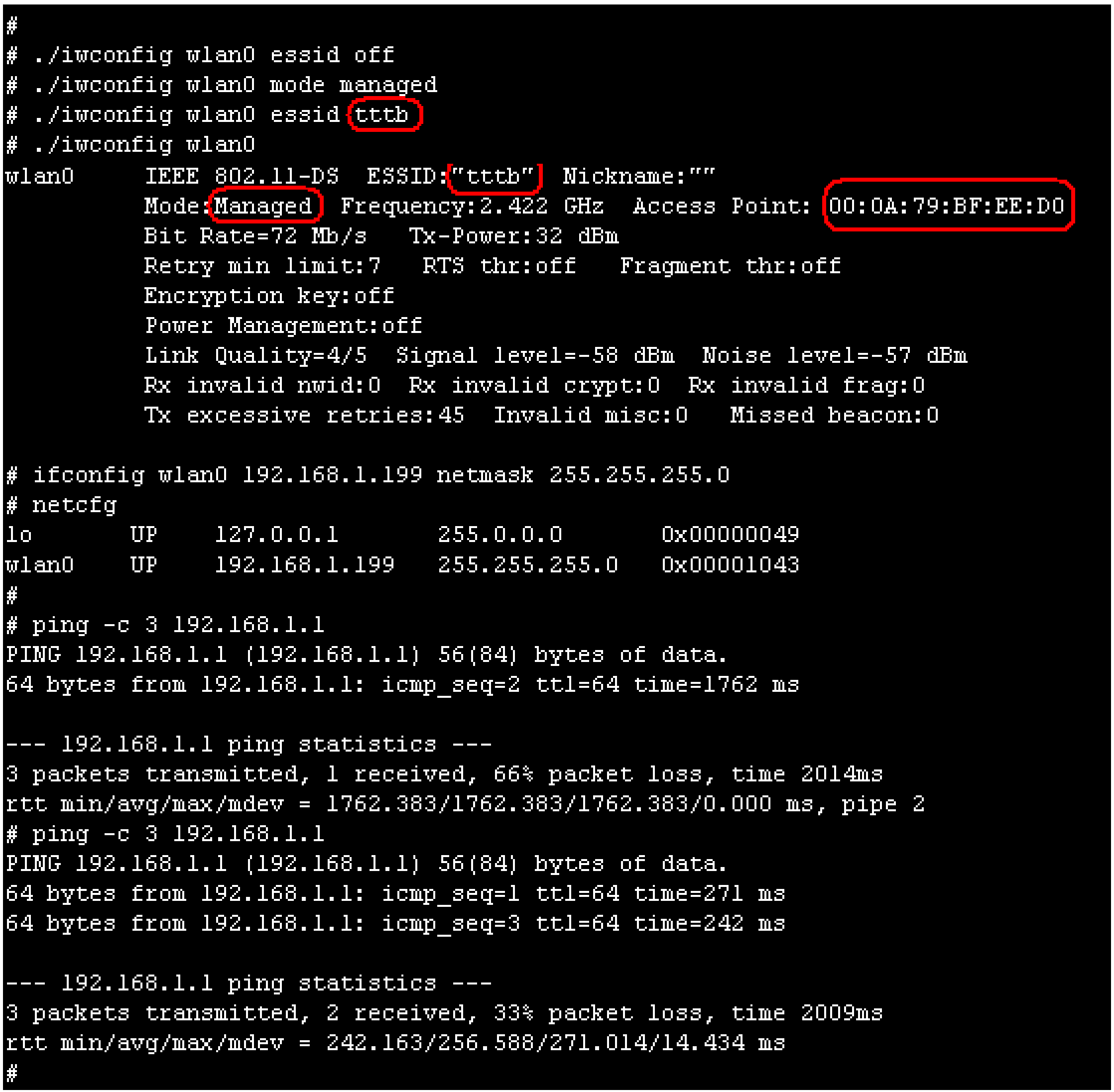


```
# ./iwlist wlan0 scan
wlan0     Scan completed :
          Cell 01 - Address: 00:0A:79:BF:EE:D0
                    ESSID:"tttb"
                    Mode:Managed
                    Frequency:2.422 GHz (Channel 3)
                    Quality:5/5  Signal level:-57 dBm  Noise level:-92 dBm
                    IE: Unknown: DD830050F204104A00011010440001021 03B0001031047
0010288028802880188 0A880000A79BFEED01021000B436F72656761204B2E4B2E1023000B43472
D574C424152474E5310240008574C424152474E531042000831323334353637381054000800060 0
50F20400011011000B43472D574C424152474E5310080002008A103C000101
                    Encryption key:off
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 9 Mb/s
                              18 Mb/s; 36 Mb/s; 54 Mb/s; 6 Mb/s; 12 Mb/s
                              24 Mb/s; 48 Mb/s
          Cell 02 - Address: 00:0A:79:BF:EE:D1
                    ESSID:"CG-Guest"
                    Mode:Managed
                    Frequency:2.422 GHz (Channel 3)
                    Quality:5/5  Signal level:-57 dBm  Noise level:-92 dBm
                    Encryption key:off
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 9 Mb/s
                              18 Mb/s; 36 Mb/s; 54 Mb/s; 6 Mb/s; 12 Mb/s
                              24 Mb/s; 48 Mb/s
          Cell 03 - Address: 00:26:87:03:F9:05
                    ESSID:"00268703F905_2nd"
                    Mode:Managed
                    Frequency:2.457 GHz (Channel 10)
                    Quality:2/5  Signal level:-74 dBm  Noise level:-92 dBm
                    Encryption key:on
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 9 Mb/s
                              18 Mb/s; 36 Mb/s; 54 Mb/s; 6 Mb/s; 12 Mb/s
                              24 Mb/s; 48 Mb/s
```

Figure 3: Scan WLAN network

CONNECT TO AP

# ./iwconfig [Interface Name] essid off // reset essid
# ./iwconfig [Interface Name] mode managed // set to infrastructure mode
# ./iwconfig [Interface Name] essid tttb // connect to tttb



```
#
# ./iwconfig wlan0 essid off
# ./iwconfig wlan0 mode managed
# ./iwconfig wlan0 essid tttb
# ./iwconfig wlan0
wlan0     IEEE 802.11-DS  ESSID:"tttb"  Nickname:""
          Mode:Managed  Frequency:2.422 GHz  Access Point: 00:0A:79:BF:EE:D0
          Bit Rate=72 Mb/s   Tx-Power:32 dBm
          Retry min limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=4/5  Signal level=-58 dBm  Noise level=-57 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:45  Invalid misc:0   Missed beacon:0

# ifconfig wlan0 192.168.1.199 netmask 255.255.255.0
# netcfg
lo       UP    127.0.0.1       255.0.0.0       0x00000049
wlan0    UP    192.168.1.199   255.255.255.0   0x00001043
#
# ping -c 3 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=1762 ms

--- 192.168.1.1 ping statistics ---
3 packets transmitted, 1 received, 66% packet loss, time 2014ms
rtt min/avg/max/mdev = 1762.383/1762.383/1762.383/0.000 ms, pipe 2
# ping -c 3 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=271 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=242 ms

--- 192.168.1.1 ping statistics ---
3 packets transmitted, 2 received, 33% packet loss, time 2009ms
rtt min/avg/max/mdev = 242.163/256.588/271.014/14.434 ms
#
```

Figure 4: Connect to WLAN AP

WLAN POWER SAVING MODE

There’re three different power saving mode settings as follows:

- PM_OFF
  - The driver is not in Power Saving mode
  - Can be activated by following command
    ./wl PM 0
- PM_MAX
  - The driver is in Maximum Power Saving mode. The driver always goes into Sleep mode and uses the PS_POLL mechanism to retrieve packets from the AP. Performance is sacrificed for maximum power savings.
  - Can be activated by following command
    ./wl PM 1
- PM_FAST
  - Fast Power Saving mode. As long as there’re active data transfers, the driver does not go into Sleep mode. After data traffic stops, driver goes into Sleep mode. This allows for power savings in IDLE times, but provides full performance when needed.
  - Can be activated by following command
    ./wl PM 2

WLAN DEEPSLEEP MODE

GB86321G can save more power in deepsleep mode through the following command sequence.

```
./wl mpc 0
./wl deepsleep 1
```

GB86321G can be waked up from deepsleep mode through the following command sequence.

```
./wl deepsleep 0
./wl mpc 1
```

WLAN OOB (OUT-OF-BAND) MODE

Host can enter sleep mode, but keep GB86321G alive. Once GB86321G receives any packets, it can wake up host through a pre-defined GPIO pin.

GPIO PIN AND POLARITY

The GPIO pin and polarity used to generate out-of-band interrupts are determined by variables downloaded to the dongle by the host driver during initialization (before starting the downloaded image). The variables are added in nvram.txt as follows:

sd_gpout - Specifies the GPIO signal to be driven as a host wake-up interrupt. Default: none.
sd_gpval - Specifies active polarity (1/0 implies active high/low). Default: active high.
sd_gpdc - Pulsing active: value is (active_ms << 16) | (inactive_ms). Default: no toggle.

Examples:

1. Active high interrupts on GPIO[0]:
   sd_gpout = 0
2. Active low interrupts (20 ms active low, 30 ms inactive high) on GPIO[0]:
   sd_gpout = 0
   sd_gpval = 0
   sd_gpdc = 0x14001E

OOB OPERATION

The commands to configure GB86321G enter/leave OOB mode are as following.

```
./dhd –i [Interface Name] sleep 1 // enter OOB
./dhd –i [Interface Name] sleep 0 // leave OOB
```

SOFTAP FUNCTIONALITY OVERVIEW

Many smart phones have high-bandwidth cellular data connections using technologies such as HSPA, UMTS, EVDO, and so on. In addition to provide access to network-based services from the phone itself, many carriers want to enable their customers to use the connection for other WLAN-enabled devices, such as computers and handheld video games.

For other devices to connect to the cellular phone, the WLAN chipset must be configured for SoftAP operation. Multiple WLAN-enabled devices can then connect to the cellular phone and share the connection. This capability is known as tethering.

Several other protocols are required, however, before the WLAN-enabled devices can successfully share a connection with the cellular phone. These protocols include DHCP (for giving associated devices their own IP address) and Network Address Translation (NAT). There protocols are layer-3 protocols that exist entirely above Ampak GB86321G SoftAP implementation.

Following table introduce the SoftAP features:

Feature	Description
Stations supported	8
Station power save support	IEEE and WMM-PS
Security	Open, WEP, WPA-PSK(TKIP), and WPA2-PSK(TKIP+AES)
WEP keys supported	4
SSID broadcast disable	Yes
Allow/deny list	Yes, through MAC address filtering
Association station list	Yes
Limit station associations	Yes, the maximum = 8

Table 1: SoftAP Features

The vast majority of the functionality required to implement a SoftAP solution is implemented in the firmware that is executed by the on-chip processor. Consequently, the software and CPU load on the host is relatively small.

SOFTAP FIRMWARE

If the target device requires SoftAP support, a firmware file with SoftAP support must be downloaded to the chipset. The firmware binary file must have an ap in the file name, such as
sdio-g-cdc-full11n-reclaim-roml-apsta-idsup-idauth.bin.

SOFTAP OPERATION

The following command sequence can be used as a template for bringing up a SoftAP on the primary WLAN interface.

```
/* Enable SoftAP mode */
./wl mpc 0
./wl down
./wl ap 1

/* Set the operating channel */
./wl channel 11

/* OR use auto-channel selection */
./wl autochannel 1      //Scan all the channels
sleep 1                 //Add some delay for channel scan
./wl autochannel 2      //Auto select channel

./wl up

/* For open authentication, no security */
./wl wsec 0
./wl wpa_auth 0

/* OR for WEP security */
./wl wsec x
./wl addwep 0 xxxxxxxxxx
./wl wpa_auth x
```

```
/* OR for WPA-Personal security */
./wl wsec x                   /* 2 for TKIP, 4 for AES, 6 for both */
./wl addwep 0 xxxxxxxxxx  /* raw 64-byte HEX PMK */
./wl wpa_auth x               /* 4 for WPA-PSK, 128 for WPA2-PSK, 132 for both
*/

/* Set maximum allowed connections */
./wl maxassoc 8
/* Set allow/deny MAC address list */
./wl macmode 1                /* 0 = disable, 1 = allow, 2 = deny */
./wl mac xx:xx:xx:xx:xx:xx [xx:xx:xx:xx:xx:xx …]

/* if desired, disable SSID broadcast */
./wl closed 0                 /* 0 = open, 1 = hidden */
./wl ssid xxxxx               /* set SSID, enable BSS */
```

BLUETOOTH SOFTWARE ARCHITECTURE OVERVIEW

BlueZ is the official Linux Bluetooth stack as well as Android. It provides support for core Bluetooth layers and protocols. We use it to provide Bluetooth profiles on GB86321G and it consists of following components (see also figure 5):

- HCI Core
- HCI UART, USB and Virtual HCI device drivers
- L2CAP protocol module
- Configuration and testing utilities

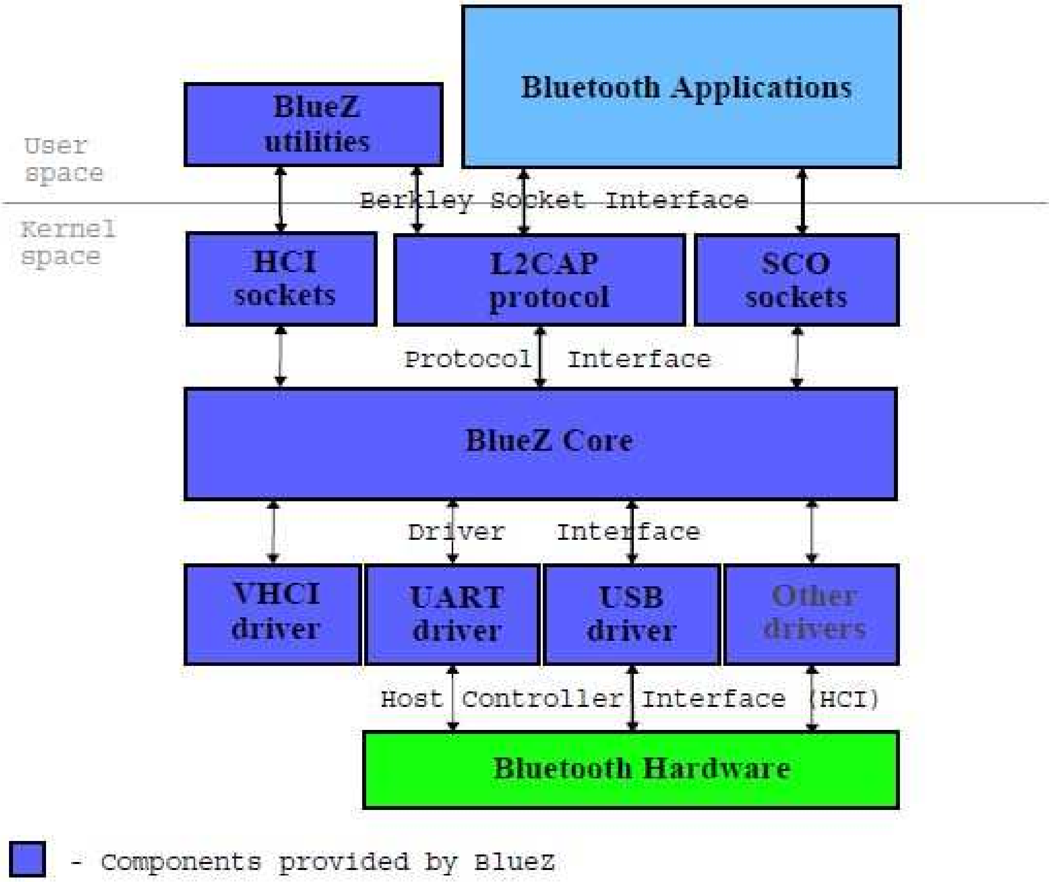


Figure 5: BlueZ Overview Diagram

In our case, we use UART as the Host Controller Interface (HCI) and GB86321G is the Bluetooth hardware in figure 5.

BLUETOOTH SOFTWARE PACKAGE

The provided Bluetooth software package contains following files:

- HCD configuration file (BCM4329B1_374.hcd)
- Hciattach program
- Hciconfig from BlueZ
- Hcitool from BlueZ
- Hciattach_test_mode program

BLUETOOTH INSTALLATION

ENABLE BLUETOOTH FUNCTION OF LINUX KERNEL

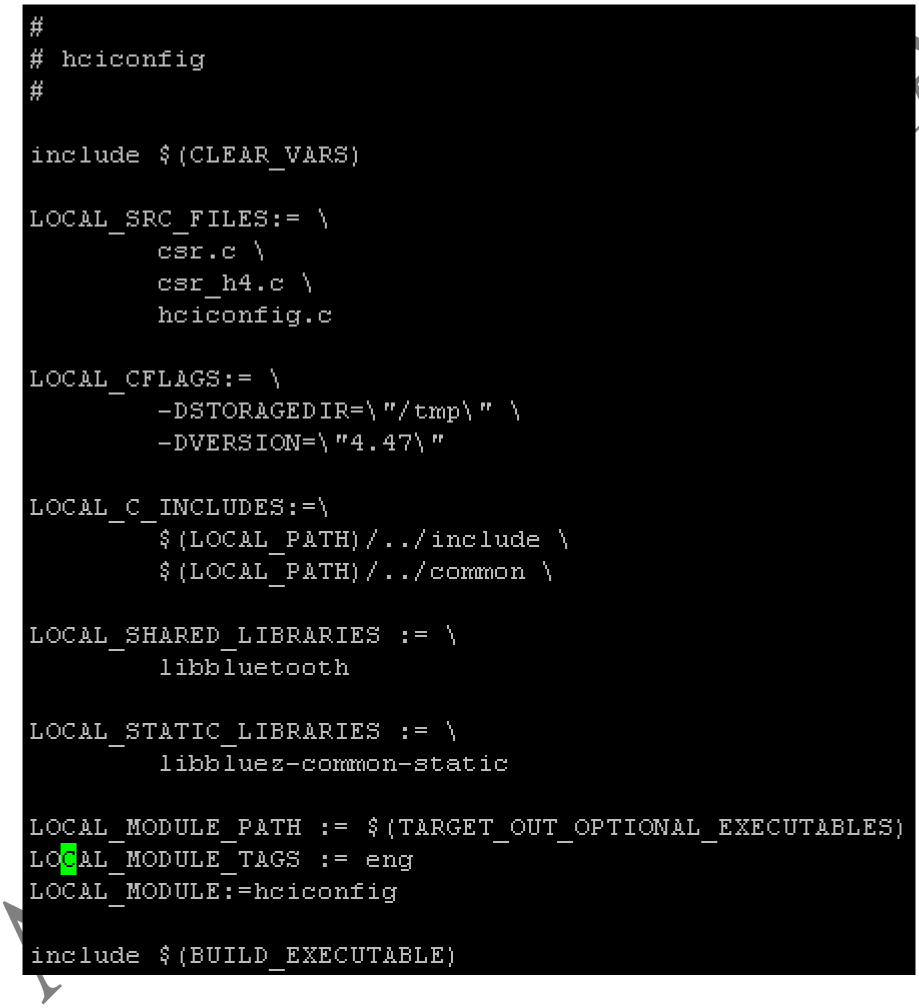
Please add following items into your kernel configuration:

```
CONFIG_BT_HCIUART=y
CONFIG_BT_HCIUART_H4=y
CONFIG_BT=y
CONFIG_BT_L2CAP=y
CONFIG_BT_SCO=y
CONFIG_BT_RFCOMM=y
CONFIG_BT_RFCOMM_TTY=y
CONFIG_BT_BNEP=y
CONFIG_BT_BNEP_MC_FILTER=y
CONFIG_BT_BNEP_PROTO_FILTER=y
CONFIG_BT_HIDP=y
```

Compile Bluetooth Protocol Stack (BlueZ) Tools

1. Please add "BOARD_HAVE_BLUETOOTH := true" in vendor/${VENDOR}/${PRODUCT}/BoardConfig.mk to compile BlueZ.

2. Please make sure your TARGET_BUILD_VARIANT is eng to or remove LOCAL_MODULE_TAGS of hciconfig and hcitool in external/bluetooth/bluez/tools/Android.mk to build the two tools.



```
#
# hciconfig
#

include $(CLEAR_VARS)

LOCAL_SRC_FILES:= \
        csr.c \
        csr_h4.c \
        hciconfig.c

LOCAL_CFLAGS:= \
        -DSTORAGEDIR=\"/tmp\" \
        -DVERSION=\"4.47\"

LOCAL_C_INCLUDES:=\
        $(LOCAL_PATH)/../include \
        $(LOCAL_PATH)/../common \

LOCAL_SHARED_LIBRARIES := \
        libbluetooth

LOCAL_STATIC_LIBRARIES := \
        libbluez-common-static

LOCAL_MODULE_PATH := $(TARGET_OUT_OPTIONAL_EXECUTABLES)
LOCAL_MODULE_TAGS := eng
LOCAL_MODULE:=hciconfig

include $(BUILD_EXECUTABLE)
```

Figure 6: Compile Bluetooth Protocol Stack (BLUEZ) Tools

3. After android been built, please make sure hciconfig and hcitool are in /system/xbin

ENABLE BLUETOOTH

1. Please put provided hciattach program in /system/xbin
2. Initialization Steps
   A.
   ```
   #hciattach -p -f BCM4329B1_374.hcd /dev/ttyAMA1bcmbt 115200 flow
   ```

   Download configuration file: BCM4329B1_374.hcd
   Serial device name: /dev/ttyAMA1
   Device: bcmbt
   Baudrate: 115200
   Flow control: flow

   B.
   ```
   #hciconfig hci0 up
   ```

3. Check Bluetooth device status
   ```
   #hciconfig
   ```

BLUETOOTH OPERATION

```
#cd /system/xbin
#hciattach -p -f BCM4329B1_374.hcd /dev/ttyAMA1bcmbt 115200 flow

start reset bcm2048
resp: 04 0e 04 01 03 0c 00
start HCI_DOWNLOAD_MINIDRIVER
      resp: 04 0e 04 01 2e fc 00
      resp[0]:34
      resp[1]:31

bcm2048 start download pram
to load HCD file BCM4329_374.hcd.

01 4c fc 2c 10 74 08 00 01 08 00 ea 44 72 42 04
18 92 70 fd 04 00 ff ff ff ff 40 06 00 00 00 00
b0 29 43 02 0a 00 38 74 08 00 00 00 00 00 00 00
//Skip some lengthy logs
```

```
.ignore 0xfc4e vs command from hcd file!
bcm2048 start HCI_LAUNCH_RAM
read_hci_event[0]=0x4
finished launch ram:
    resp: 04 0e 04 01 4e fc 00
bcm2048 start Write_Voice_Setting
    resp: 04 0e 04 01 26 0c 00
bcm2048 start Write_SCO_PCM_Int_Param
    resp: 04 0e 04 01 1c fc 00
bcm2048 start Write_PCM_Data_Format_Param
    resp: 04 0e 04 01 1e fc 00
bcm2048 start Update_UART_Baud_Rate
Baud rate parameters: 115200
    resp: 04 0e 04 01 18 fc 00
start Write_Scan_Enable bcm2048
    resp: 04 0e 04 01 1a 0c 00
Endof to init_uart()
Device setup complete
pid : 1737
#
#hciconfig hci0 up              //Activate Bluetooth interface
#
# hciconfig
hci0:     Type: UART
BD Address: 43:29:B0:00:00:00 ACL MTU: 1021:7 SCO MTU: 64:1
UP RUNNING
RX bytes:352 acl:0 sco:0 events:10 errors:0
TX bytes:45 acl:0 sco:0 commands:10 errors:0
#
#hcitool scan              //Perform site survey
Scanning ...
          00:22:43:A0:A7:0A        n/a
          00:10:60:56:56:7B        hhhh
          00:1A:6B:85:F3:67        n/a
          00:22:43:A0:A7:48        AmUrO
          00:1F:E1:E1:A1:8F        GEMTEK-8AE51F68
```

BLUETOOTH POWER SAVING MODE

The GB86321G Bluetooth supports a special Sleep Mode to reduce power consumption. The Sleep Mode is **DISABLED** in firmware by default and must be enabled by the host through following command.

SOFTWARE COMMAND FOR ENABLE SLEEP MODE

#./hcitool cmd A B C D E F G H I J K

Parameter	Description
A	(ogf) must be 0x3F
B	(ocf) must be 0x0027
C	Sleep_Mode (1 bytes) 0x00: No Sleep Mode 0x01: UART Sleep Mode 0x02: UART Sleep Mode with messaging 0x03: USB Sleep Mode 0x05: USB Sleep Mode with Host Wake
D	Idle_Threshold_Host(1 bytes) 0xXX: Host Idle Threshold, applicable to Sleep Mode 1, 2, 5. This is the number of firmware loops executed with no activity before the Host Wake line is deasserted. Activity includes HCI traffic excluding certain sleep mode commands and the presence of SCO connection if the “Allow Host Sleep During SCO” flag is not to set 1. Each count of this parameter is roughly equivalent to 300 ms. For example, when the parameter is set to 16 (0x10), the Host wake line will be deasserted after approximately 4.8 seconds of inactivity.
E	Idle_Threshhold_HC (1 byte) 0xXX: Host Control Idle Treshod, applicable to Sleep Mode 1, 2, 3, 5. This is the number of firmware loops executed with no activity before the HC is considered idle. Depending on the mode, HC may then attempt to sleep. Activity includes HC traffic excluding certain sleep mode commands and the presence of ACL/SCO connections. Each count of this parameter is roughly

	equivalent to 300 ms. when the parameter is set to 16 (0x10), the HC will be considered after approximately 4.8 seconds of inactivity.
F	GPIO_0_Active_Mode(1 byte) 0x00: Active Low 0x01: Active High
G	GPIO_3_Active_Mode (1 byte) 0x00: Active Low 0x01: Active High
H	Allow_Host_Sleep_During_SCO (1 byte) 0x00-0x01: Applicable to Sleep Mode 1, 2, 3, 5. When this flag is set to 0, the host is not allowed to sleep while an SCO connection is active. In modes 1 and 2, the device will keep the host wake line asserted while an SCO connection is active. In mode 3, the device will immediately issue a USB RESUME if the host issues a SUSPEND. When this flag is set to 1, the host can sleep while an SCO is active. This flag should only be set to 1 if SCO traffic is directed to the PCM interface.
I	Combine_Sleep_Mode_And_LPM (1bytes) 0x00-0x01: Applicable to Sleep Mode 1, 2, 3, 5. In mode 0, always set byte 7 to 0. In all sleep modes, device always requires permission to sleep between scans / periodic inquiries regardless of the setting of this byte. In modes 1 and 2, if the byte is set, device must have “permission” to sleep during the low power modes of sniff, hold, and park. If byte is not set, device can sleep without permission during these modes. Permission to sleep mode 1 is obtained if the BT_WAKE signal is not asserted. Permission to sleep mode 2 occurs after the Sleep Request / Sleep Reguest ACK exchange. In modes 3 and 5, if the byte is set to 0, the device will not be able to sleep during the lower power modes. If it is set to 1, the device will be able to sleep during the lower power modes.
J	Enable_Tristate_Control_Of_UART_Tx_Line (1bytes) 0x00-0x01: Applicable to Sleep Mode 1 and 2. When set to 0, the device will not tristate its UART TX line before going to sleep.

	When set to 1, the device will tristate its UART TX line before going to sleep.
K	Active_Connection_Handling_On_Suspend(1bytes) 0x00-0x01: Suspend Behavior, applicable to modes 3 and 5.

Table 2: Bluetooth Sleep Mode Command Parameters

WAKE UP FROM SLEEP MODE

The Bluetooth can be woken from sleep mode only by the below two methods.

1. The host assert BT_WAKE pin
2. The remote Bluetooth device communicates with it via radio

OPERATION

Following is an example showing how to enter sleep mode and wake up.
Let's assume BT_Wake pin connects to the GPIO pin 45 of the host CPU.

```
#gpio get 45
0

#./hcitool cmd 0x3f 0x0027 0x01 0x0 0x0 0x1 0x0 0x0 0x0 0x0 0x0

< HCI Command: ogf 0x3f, ocf 0x0027, plen 9
   01 00 00 01 00 00 00 00 00
> HCI Event: 0x0e plen 4
   01 27 FC 00
#./hcitool scan
Scanning ...
Inquiry failed: Connection timed out
#gpio set 45 1
#./hcitool scan
Scanning ...
      00:1E:45:E3:9A:0C    K800i
      F0:7B:CB:A8:86:52    BEN-99
      00:15:83:36:18:9F    andy-desktop-0
      00:1F:E1:E1:A1:8F    GEMTEK-8AE51F68
# gpio set 45 0
```

```
#./hcitool scan
Scanning ...
Inquiry failed: Connection timed out
#gpio set 45 1
#./hcitool scan
Scanning ...
        F0:7B:CB:A8:86:52     BEN-99
        00:15:AF:FD:4A:7D     MYPC-E180EB2C24
        00:1C:26:EB:30:32     COCO-PC
        00:1E:45:E3:9A:0C     K800i.
        00:1F:E1:E1:A1:8F     GEMTEK-8AE51F68
#
```

BLUETOOTH TEST MODE

TEST EQUIPMENT

Anritsu MT8852B is used for Bluetooth testing.

OPERATION

```
#./hciattach_test_mode -p -f BCM4329B1_374.hcd /dev/ttyAMA1 bcmbt
115200 flow
start reset bcm2048
            resp: 04 0e 04 01 03 0c 00
bcm2048 start HCI_DOWNLOAD_MINIDRIVER
            resp: 04 0e 04 01 2e fc 00
resp[0]:34
resp[1]:31

bcm2048 start download pram
to load HCD file BCM4329B1_374.hcd.
start Write_Scan_Enable bcm2048
            resp: 04 0e 04 01 1a 0c 00
start   Set Event Filter bcm2048
            resp: 04 0e 04 01 05 0c
start Enable_Device_Under_Test_Mode bcm2048
            resp: 04 0e 04 01 03 18 00
Device setup complete
pid : 2512
#
```

WRITE MAC ADDRESS

Programmers have two ways to write Bluetooth MAC address. One is read MAC address from a file. The other is from command parameter. If there is no one used, the default MAC address is stored. If both of them are used, the first way reading MAC address from a file gets high priority.

READ MAC ADDRESS FROM A FILE

```
./hciattach -p -f BCM4329_374.hcd -m /system/bin/mac_addr.txt /dev/ttyAMA1 bcmbt 1152000 flow

start reset bcm2048
        resp: 04 0e 04 01 03 0c 00
bcm2048 start HCI_DOWNLOAD_MINIDRIVER
        resp: 04 0e 04 01 2e fc 00
resp[0]:34
resp[1]:31

bcm2048 start download pram
to load HCD file BCM4329_374.hcd.
      .
      //Skip some lengthy logs
      .
finished launch ram:
        resp: 04 0e 04 01 4e fc 00
bcm2048 start Write_Voice_Setting
        resp: 04 0e 04 01 26 0c 00
bcm2048 start Write_SCO_PCM_Int_Param
        resp: 04 0e 04 01 1c fc 00
bcm2048 start Write_PCM_Data_Format_Param
        resp: 04 0e 04 01 1e fc 00
Read MAC Address file and prepare to write MAC address
bcm2048 start Write_BD_ADDR 00:15:56:5A:1E:89
        resp: 04 0e 04 01 01 fc 00
```

```
bcm2048 start Update_UART_Baud_Rate
Baud rate parameters: 1152000
        resp: 04 0e 04 01 18 fc 00
start Write_Scan_Enable bcm2048
        resp: 04 0e 04 01 1a 0c 00
Device setup complete
pid : 3760
#
# hciconfig hci0 up
# hciconfig
hci0:      Type: UART
                BD Address: 00:15:56:5A:1E:89 ACL MTU: 1021:7 SCO MTU: 64:1
                UP RUNNING
                RX bytes:1492 acl:0 sco:0 events:24 errors:0
                TX bytes:108 acl:0 sco:0 commands:17 errors:0
# cat /system/bin/mac_addr.txt
00:15:56:5A:1E:89
#
```

From Command Parameter

```
# ./hciattach -p -f BCM4329_384.hcd /dev/ttyAMA1 bcmbt 1152000 flow
00:12:34:56:78:90
start reset bcm2048
        resp: 04 0e 04 01 03 0c 00
bcm2048 start HCI_DOWNLOAD_MINIDRIVER
        resp: 04 0e 04 01 2e fc 00
resp[0]:34
resp[1]:31

bcm2048 start download pram
to load HCD file BCM4329_374.hcd.
      .
      //Skip some lengthy logs
      .
finished launch ram:
        resp: 04 0e 04 01 4e fc 00
bcm2048 start Write_Voice_Setting
        resp: 04 0e 04 01 26 0c 00
bcm2048 start Write_SCO_PCM_Int_Param
        resp: 04 0e 04 01 1c fc 00
bcm2048 start Write_PCM_Data_Format_Param
        resp: 04 0e 04 01 1e fc 00
bcm2048 start Write_BD_ADDR 00:12:34:56:78:90
        resp: 04 0e 04 01 01 fc 00
bcm2048 start Update_UART_Baud_Rate
Baud rate parameters: 1152000
        resp: 04 0e 04 01 18 fc 00
start Write_Scan_Enable bcm2048
        resp: 04 0e 04 01 1a 0c 00
Device setup complete
pid : 4229
#
# hciconfig hci0 up
# hciconfig
hci0:Type: UART
        BD Address: 00:12:34:56:78:90 ACL MTU: 1021:7 SCO MTU: 64:1
        UP RUNNING
        RX bytes:352 acl:0 sco:0 events:10 errors:0
        TX bytes:45 acl:0 sco:0 commands:10 errors:0
#
```

WLAN AND BLUETOOTH COEXISTENCE ENHANCEMENT

GB86321G implements the highly sophisticated Enhanced Collaborative Coexistence radio coexistence algorithm and hardware mechanism, allowing for an extremely collaborative Bluetooth coexistence scheme along with coexistence support for WLAN.
Moreover, you can also use following method to enhance WLAN and Bluetooth coexistence.

MITIGATE BT MUSIC DISCONTINUOUSLY WHEN WLAN ENABLED

When you listen to music through BT and browse internet through WLAN concurrently. You can use following steps to enhance the coexistence.

1. Connect to BT headset
2. Get handler through hcitool

```
# hcitool con
Connections:
        < ACL 00:15:08:12:21:80 handle 12 state 1 lm SLAVE AUTH
ENCRYPT   mtu 0 credits 0/0
```

3. Send hcitool command

```
# hcitool cmd 0x3f 0x57 0x0c 0x00
< HCI Command: ogf 0x3f, ocf 0x0057, plen 2
  0C 00
> HCI Event: 0x0e plen 4
  01 57 FC 00
```

ELIMINATE WLAN DISCONNECTED WHEN BT SCANNING

When GB86321G performs BT scanning, you can use following steps to enhance the coexistence.

1. Send hciconfig command

```
# hciconfig -a [Interface Name] pageparms 256:2048
```

2. Active BT scanning
3. Back to original parameter after BT scan devices completely.

```
# hciconfig -a [Interface Name] pageparms 18:2048
```